Update-Programmierung von Kraftfahrzeugen über heterogene Kommunikationsnetzwerke

Von der Fakultät für Elektrotechnik und Informationstechnik

der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung der Würde

eines Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Dissertation

von: Ralf Schade

aus Erfurt

eingereicht am: 12. Juli 2006

mündliche Prüfung am: 06. Oktober 2006

Vorsitzender: Prof. Dr.-Ing. Thomas Form

Referenten: Prof. Dr.-Ing. Jörn-Uwe Varchmin

Prof. Dr.-Ing. Bernard Bäker

2006

Vorwort

Die vorliegende Arbeit enstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik (emg) der Technischen Universität Carolo-Wilhelmina zu Braunschweig. Das Projekt wurde seitens der Konzernforschung der Volkswagen AG unterstützt, deren Mitarbeitern Thomas Engel und Jens Ende ich für ihre Hilfe danken möchte.

Mein besonderer Dank gilt Prof. Dr.-Ing. Jörn-Uwe Varchmin, auf den die Bedeutung des Wortes „Doktorvater“ in höchstem Maße zutreffend ist. Er verstand es zu jeder Zeit, mir bei schwierigen Fragen in technischer und moralischer Hinsicht Unterstützung zu geben und damit die Motivation zur Beendigung der Arbeit aufrecht zu halten.

Bei Herrn Prof. Dr.-Ing. Bernard Bäker möchte ich mich herzlich für die Übernahme des zweiten Berichtes bedanken. Ebenso danke ich Prof. Dr.-Ing. Thomas Form für den Vorsitz der Promotionskomission und seine angenehme Art und Weise, durch das Promotionsverfahren zu führen.

Besonderer Dank gilt meinen Eltern, die mir meine wissenschaftliche Ausbildung ermöglichten und mich zu jedem Zeitpunkt förderten. Ebenfalls danken möchte ich meinem Bruder Frank, der an einem Teil dieser Arbeit mitwirkte und mich fachlich und unterstützend begleitete. Meine Verlobte Susanne Wandel hat durch ihren sanften Druck und seelischen Rückhalt die Fertigstellung der Arbeit vorangetrieben. Für ihr Verständnis während dieser Zeit danke ich ihr.

Ich konnte mich immer auf meinen Freundeskreis verlassen, die ihre Beiträge in Form von Korrekturlesen, Anmerkungen und Diskussionen brachten. Guido Brunken und Franck Rogge haben mir durch fachliche Unterstützung und mit viel konstruktiver Kritik bei der Verwirklichung dieser Arbeit geholfen. Darüber hinaus korrigierten sie zusammen mit Tanja Düring und Matthias Thieme diese Arbeit.

Meinen Kollegen möchte ich für die sehr angenehme und freundschaftliche Arbeitsatmosphäre danken, in dem diese Arbeit entstand. Durch ihre Bereitschaft zu Hilfe und Diskussionen halfen sie beim Gelingen der Arbeit. Ebenfalls sehr wichtig für den Fortschritt dieser Arbeit waren die Studenten, die im Rahmen ihrer Studien- und Diplomarbeiten einen großen Anteil an der Realisierung der gesteckten Ziele hatten. Vielen Dank dafür.

Inhaltsverzeichnis

Abkürzungen

Begriffe

ABS	Anti-Blockier-System
ACC	Adaptive Cruise Control (dt. ADR)
ADR	Automatische Distanz Regelung (engl. ACC)
ADU	Analog/Digital-Umsetzer
AEG	Automotive Expert Group
AP	Access Point
API	Application Programming Interface
ARP	Adress Resolution Protocol
ATM	Asynchronous Transfer Mode
BAP	Bedien- und Anzeige Protokoll
BLF	Bootloader Flash
CAN	Controller Area Network
CARB	California Air Resources Board
CAS	Collision Avoidance Symbol
CCC	Car to Car Communication
CD	Compact Disc
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
CWG	Car Working Group
DAB	Digital Audio Broadcasting
DAB	Digital Audio Broadcast
DAU	Digital/Analog-Umsetzer
DDP	Data Display Protocol
DSP	Digital Signal Processing / Processor
DSRC	Dedicated Short Range Communication
DVB-T	Digital Video Broadcast – Terrestical
DVD	Digital Versatile Disc / Digital Video Disc
ECU	Electronic Control Unit
EDGE	Enhanced Datarate for Global Evolution
EDR	Enhanced Data Rate
EEPROM	Electrically Erasable Programmable Read-Only Memory

EMV	Elektromagnetische Verträglichkeit
EPROM	Erasable Programmable Read-Only Memory
ESP	Elektronisches Stabilitätsprogramm
FDMA	Frequency Division Multiple Access
FIFO	First In / First Out
FIT	Failure In Time
FP	Fahrprogramm
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
GW	Gateway
HCI	Host Controller Interface
HSCSD	High Speed Circuit Switched Data
IC	Integrated Circuit
IP	Internet Protocol
IPC	Inter Process Communication
ISR	Interupt Service Routine
IVT	Interupt Vector Table
Kfz	Kraftfahrzeug
KWP	Key Word Protocol
L2CAP	Logical Link Control and Adaption Protocol
LCS	Lane Change Support (dt. SWA)
LDW	Lane Departure Warning (dt. SPA)
LIN	Local Interconnect Network
LMP	Link Manager Protocol
LWL	Lichtwellen-Leiter (engl. POF)
MOST	Media Oriented Systems Transport
MP3	Moving Picture Expert Group 1.0 Layer 3 (Musikkompressionsverfahren)
MTS	Media Access Test Symbol
NAND	Nicht UND–Verknüpfung (Ausdruck der Boolschen Algebra)
NIT	Network Idle Time
NM	Network Management
NOR	Nicht ODER–Verknüpfung (Ausdruck der Boolschen Algebra)
OBD	On Board Diagnose
OEM	Original Equipment Manufacturer
OSEK	Offene Systeme und deren Schnittstellen für die Elektronik im Kfz
OSI	Open Systems Interconnection

PCI	Peripheral Component Interconnect
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
POF	Polymer Optical Fiber (dt. LWL)
PPM	Parts Per Million
PPP	Point-to-Point Protocol
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RDS	Radio Data Service
RFID	Radio Frequency Identification
RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computing
RKE	Remote Keyless Entry
ROM	Read Only Memory
SAE	Society of Automotive Engineers
SG	Steuergerät
SIG	Special Interest Group
SIP	Systemintegritätsprotokoll
SNMP	Simple Network Management Protocol
SPA	Spurassistent (engl. LDW)
SWA	Spurwechselassistent (engl. LCA)
TCP	Transmission Control Protocol
TCS	Telephony Control Protocol
TDMA	Time Division Multiple Access
TMC	Traffic Message Channel
TP	Transportprotokoll
TTA	Time Triggered Architecture
TTL	Transistor-Transistor-Logik ($V_{IL} \leq 0{,}8V$, $V_{IH} \geq 2{,}0V$)
TTP	Time Triggered Protocol
UDP	User Datagram Protocol
UDS	Unified Diagnostic Services
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
UWB	Ultra Wideband
VAN	Vehicle Area Network
VDX	Vehicle Distributed eXecutive
WEP	Wired Equivalent Privacy
W-LAN	Wireless Local Area Network

WPA	Wireless Fidelity Protected Access
WPAN	Wireless Personal Area Network
WUS	Wake Up Symbol

Firmen und Institutionen

ARM	Advanced RISC Machines
Audi	Audi AG
BMW	Bayerische Motoren Werke AG
Bosch	Robert Bosch GmbH
DC	Daimler Chrysler AG
GM	General Motors
IEEE	Institute of Electrical and Electronical Engineers
ISO	International Organisation for Standardisation
RTAI	Real Time Application Interface
SAE	Society of American Engineers
VW	Volkswagen AG

1 Einleitung

So wie sich unsere Gesellschaft, angetrieben durch Innovationen und Technologien, von einer Industriegesellschaft zu einer Informationsgesellschaft entwickelt hat, vollzieht sich dieser Wandel seit einigen Jahren am Automobil. Was ursprünglich als eine Domäne des Maschinenbaus begann, wird heute mehr und mehr durch Elektrotechnik und zukünftig durch Informationstechnik geprägt. Die Gründe für diese Entwicklung liegen in den vielfältigen Anforderungen, die an heutige Kraftfahrzeuge gestellt werden (siehe Abbildung 1). Die Kunden wünschen sich Leistung und Komfort zu akzeptablen Preisen. Sie erwarten von der Automobilindustrie, dass sie den schnellen Entwicklungszyklen der Mikroelektronik- und Kommunikationsindustrie folgt, was bei den langen Entwicklungs- und Produktlebenszyklen eines Kraftfahrzeugs eine große Herausforderung darstellt. Die Globalisierung der Märkte setzt ein hohes Maß an Flexibilität bei der Fahrzeugentwicklung voraus, um den unterschiedlichsten länderspezifischen Standards genügen zu können. Der Gesetzgeber fordert durch immer strengere Abgasverordnungen umweltschonende und sparsame Fahrzeuge. Darüber hinaus besteht für die Automobilindustrie auch aus eigenem Interesse die Pflicht, die Sicherheit und den Komfort beim Fahren ständig zu verbessern. Erreicht werden soll dies unter anderem mit Assistenzsystemen, die verstärkt in neuen Fahrzeugmodellen zum Einsatz kommen. Die Erfüllung dieser Anforderungen wird durch die Kraftfahrzeugelektronik erst ermöglicht, die dabei sehr stark von den Innovationen der Halbleiter- und Kommunikationsindustrie profitiert [Schade05].

Abbildung 1: Anforderungen an moderne Kraftfahrzeuge

Gleichzeitig wächst im Automobil mit zunehmender Komplexität das Risiko der Anfälligkeit für Ausfälle bis zu vollständigem Versagen der neuen Technik, was Schlagzeilen wie „Elektronik – Pannenursache Nr.1“, „Softwarefalle Kfz“ oder auch die Entwicklung der Fahrzeug-Rückrufe beweisen (Abbildung 2). Neben der

Innovationsfähigkeit wird daher künftig auch die Beherrschbarkeit der wachsenden Systemkomplexität zum zentralen Erfolgsfaktor im Wettbewerb unter den Herstellern. Neue Ideen und Technologien werden in Zukunft oftmals auf Basis von vernetzten und komplexen Systemen entstehen [Reichart05]. Das Fahrzeug wird daher nicht mehr als ein in sich abgeschlossenes System behandelt werden können, sondern muss definierte Schnittstellen nach „außen“ bereitstellen, um Diagnose, Update-Programmierung oder den Austausch beliebiger Informationen zwischen den Systemen zu ermöglichen. Weitere Aufgaben, wie eine verstärkte Integration von Softwaremodulen unterschiedlicher Lieferanten in ein Steuergerät, die Update- oder Upgrade-Fähigkeit der Fahrzeuge über die gesamte Produktlebenszeit und neue zeitgesteuerte Datenbussysteme, erzwingen ein Überdenken heutiger Systemdesigns und Integrationskonzepte.

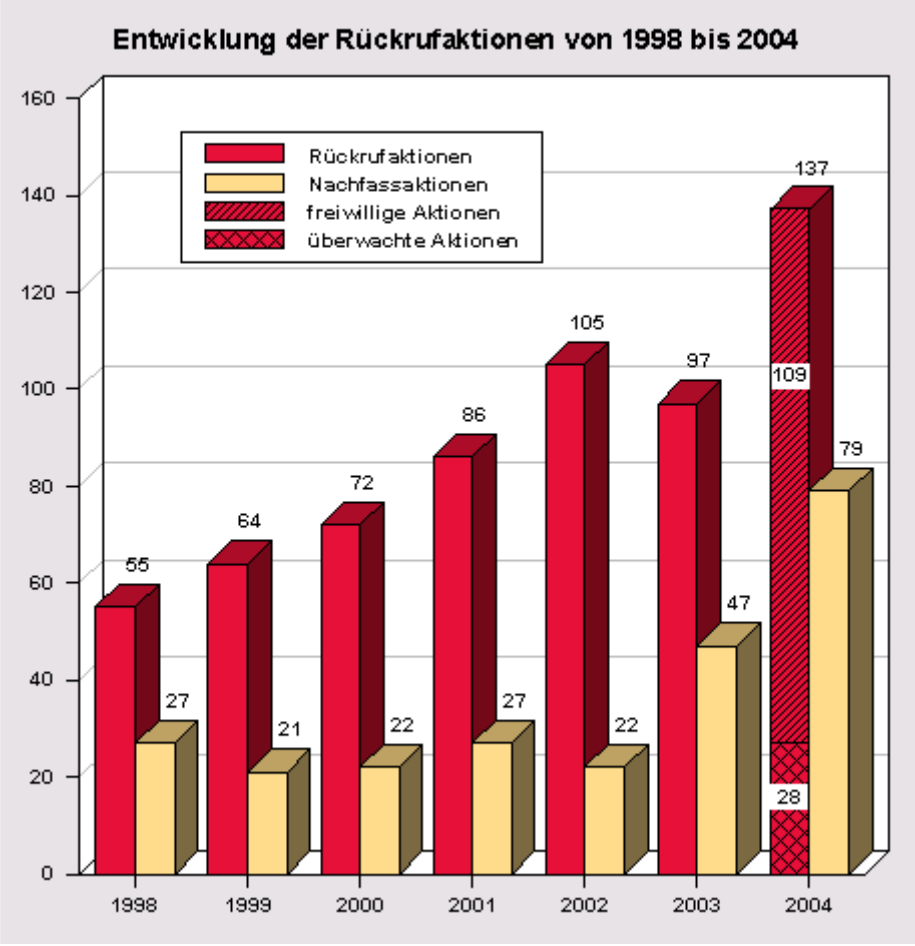

Abbildung 2: Entwicklung der Rückrufaktionen (Quelle: Kraftfahrtbundesamt)

Die vorliegende Arbeit widmet sich der Aufgabe, dem steigenden Bedarf nach Vernetzung und Austausch von Informationen mit einer geeigneten Kommunikationsarchitektur zu begegnen, deren Mittelpunkt das zeitgesteuerte Datenbussystem FlexRay bildet. Sie überprüft deren Anforderungen bezüglich der neuen zu verwendenden Kommunikationsprotokolle, die sich vom bislang eingesetzten Prinzip des ereignisgesteuerten Datenaustauschs lösen und sich mit den gängigen Werkzeugen schwer oder gar nicht darstellen lassen.

Eine wichtige Schnittstelle für den Zugriff von außen wird dabei der drahtlose Diagnosezugang sein. Dabei wird als Anwendungsfall das Flashen (Update-Programmierung) von Steuergeräten genauer betrachtet werden, welcher hohe Ansprüche an das Zeit- und Übertragungsverhalten stellt und bedingt durch den steigenden Softwareanteil im Kraftfahrzeug weiter an Bedeutung zunehmen wird.

Das Ziel dieser Arbeit ist es, den Nachweis zu erbringen, dass ein für das Kommunikationsnetz im Automobil neuer Backbone-Architekturansatz mit FlexRay als schnellem Datenbus die aufkommenden Probleme der Kommunikationsengpässe vermeiden kann. Als besonders kritischer Anwendungsbereich – weil hier sehr hohe Zuwachsraten in den Datenmengen prognostiziert werden – wird die „Update-Programmierung" ausgewählt. Die Schwerpunkte liegen dabei auf der Untersuchung der notwendigen Voraussetzungen hinsichtlich der Steuergeräteprogrammierung über eine heterogene Fahrzeug-Netzwerkarchitektur. Insbesondere die Vorteile der Architektur für den schnellen Datentransport ins Fahrzeug durch die optimale Bandbreitenausnutzung des Backbones mit der Möglichkeit der parallelen Programmierung von Steuergeräten soll nachgewiesen werden. Des Weiteren wird mit einem drahtlosen Diagnosezugang für die Kommunikationsarchitektur der zukünftigen Entwicklung bei der Fahrzeugvernetzung Rechnung getragen.

1.1 Übersicht der Arbeit

Über die historische Entwicklung der Kraftfahrzeugelektronik, die ausführlich in Kapitel 2 rekapituliert wird, leiten sich Beschränkungen der bestehenden Kommunikationsarchitekturen und Anforderungen an eine zukünftige Systemarchitektur ab, die diese Arbeit motiviert haben. Die historisch gewachsene grundlegende Systemarchitektur wird zusammen mit einer Übersicht über den aktuellen Stand der Kraftfahrzeugelektronik heutiger Automobile im Kapitel 3 vorgestellt. Während dabei der Fokus auf der Architektur und den verwendeten Datenbusse liegt, beschäftigt sich Kapitel 4 mit der Applikation „Flashen“. Dazu werden eine Einschätzung über die Perspektiven des Flash-Speichers im Automobil und eine Übersicht über die zur Update-Programmierung notwendigen Protokolle gegeben. Die daraus resultierenden Erkenntnisse wurden für den Aufbau eines Labor-Kommunikationsnetzwerks genutzt, welches im Kapitel 5 beschrieben ist. Für die Realisierung eines funktionierenden Programmierungsprozesses innerhalb dieses Labornetzwerks waren Anpassungen an Protokollen und Software notwendig, die das Kapitel 6 näher erläutert. Das Kapitel 7 fasst die Ergebnisse der Untersuchungen zusammen und gibt einen Ausblick auf die zukünftige Entwicklung von Kommunikationsnetzwerken in Fahrzeugen.

2 Kraftfahrzeugelektronik – die Geschichte der Zukunft

Eine der am häufigsten genannten Aussagen zur automobilen Zukunft ist die folgende:

„90% aller Innovationen im Automobilbereich sind Elektroniksysteme oder elektronisch gesteuerte Komponenten. Etwa 80% davon entfallen auf Software."

Diese Behauptung findet man sinngemäß bei allen großen Automobilherstellern wieder, z.B. [Dudenhöff.03], [Hofman01], [Lange03], [Pasemann03], [Teepe04], [Scharnhorst04]. Die Prozentzahlen in den einzelnen Quellen variieren leicht, die Kernaussage bleibt jedoch gleich. Der Anteil der Elektronik in Hard- und Software wird im Kraftfahrzeug weiter zunehmen, die Komplexität weiter stark steigen [Abbildung 3], [Schleuter02].

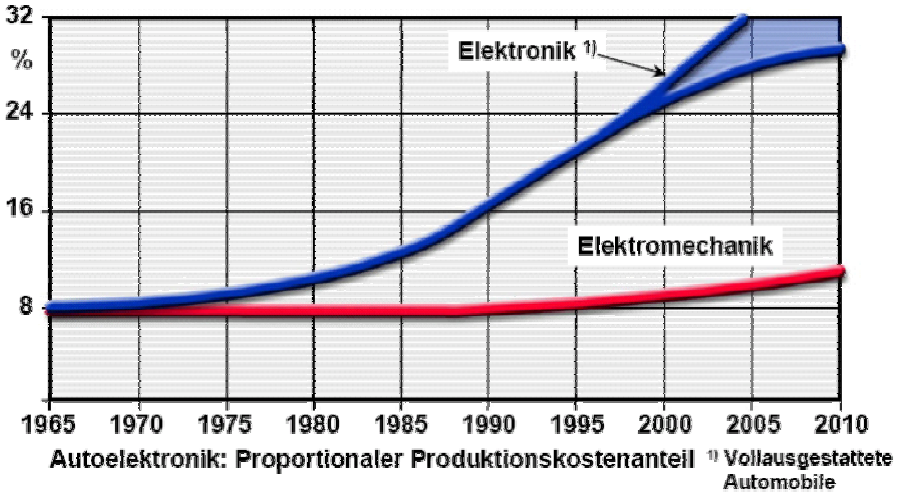

Abbildung 3: Anteil der Elektronik an den Produktionskosten [Rech04]

Wie kann diese Entwicklung begründet werden und welche Schlussfolgerungen ergeben sich daraus? Dieses Kapitel wird versuchen, Antworten auf diese Fragen zu geben, in dem es genauer auf die Entwicklung der Kraftfahrzeugelektronik eingeht. Anhand der historisch gewachsenen Elektronikstrukturen im Kraftfahrzeug lassen sich die aktuellen Probleme aufzeigen, die zukünftige Neuentwicklungen hemmen können. Dabei werden die technischen und gesellschaftlichen Triebkräfte untersucht, die einen Großteil der aktuellen Innovationen motiviert haben. Darauf aufbauend erfolgt eine Zukunftsabschätzung der Kraftfahrzeugelektronik, die wiederum Auswirkungen auf das Gesamtprodukt „Kraftfahrzeug" besitzt. Mit dieser Abschätzung ist die Grundlage für die vorliegende Arbeit geschaffen, die untersucht, wie eine zukünftige Kommunikationsarchitektur im Fahrzeug aussehen kann. Für einen detaillierten Überblick über den aktuellen Stand der Technik mit dem Schwerpunkt der Datenkommunikation im Kfz wird auf Kapitel 3 verwiesen.

2.1 Historie des Kraftfahrzeugs

Bis in das 18. Jahrhundert gab es als Fortbewegungsmittel nur die Kutsche und das Pferd. Mit der Erfindung der Eisenbahn konnte später zwar die Geschwindigkeit und damit die Mobilität gesteigert werden, man war aber an Fahrpläne und feste Haltepunkte gebunden. Erst das Automobil brachte eine universelle, schnelle und vor allem individuelle Mobilität, die auch für Privatpersonen erschwinglich wurde [Lay94]. Es ermöglicht einen flexiblen Transport von Personen und Waren über große Entfernungen. War das Automobil in seinen Anfängen noch mehr Statussymbol als ernstzunehmende Konkurrenz zu den etablierten Verkehrsträgern, so verdrängte es mit seinen Eigenschaften diese immer mehr und ist jetzt schon seit Jahrzehnten das Hauptverkehrsmittel auf der Welt.

Abbildung 4: Mercedes 40HP Simplex von 1906 [DCH]

Die meisten der heute produzierten Kraftfahrzeuge basieren auf der Grundkonstruktion des Mercedes Simplex[1] von 1906 [Abbildung 4]. Sie besitzen einen Motor vorn, ein Getriebe und Antriebswellen zu den angetriebenen Rädern. Für den Antrieb gab es im Laufe der Entwicklungsgeschichte verschiedenste Konzepte, wobei sich der Hubkolbenmotor als Antriebsquelle (Diesel oder Benzin) durchgesetzt hat. Andere Konzepte wie Wankelmotor oder Elektromotor führen auf Grund ihrer Leistungsfähigkeit, Bedienungsfreundlichkeit, technischer Einschränkungen oder fehlender Energiespeicher derzeit nur ein Nischendasein. Erst in den letzten Jahren wird der Kombination verschiedener Konzepte in einem Fahrzeug, den so genannten Hybridantrieben, eine größere Verbreitung in der Zukunft vorausgesagt. Bestes Beispiel dafür ist die Kombination Hubkolbenmotor mit Elektroantrieb, deren Entwicklung derzeit von Toyota stark vorangetrieben wird [Engine04]. Aber auch andere Konzepte in diese Richtung, wie z.B. die Brennstoffzellentechnologie, werden weiter erforscht [Winterkorn05].

Alle neuen Konzepte konnten erst mit Hilfe der modernen Elektronik realisiert werden. Das Gleiche gilt auch für die meisten der heutigen Fahrer- und

[1] Der Begriff Simplex soll auf Kaiser Wilhelm II. zurückgehen, der sich 1906 auf einer Automobilausstellung in Berlin den Startvorgang des Mercedes erklären ließ und den rund 10-minütigen „Startvorgang“ als „Simplex“ bezeichnete.

Fahrzeugassistenzsysteme, die das Fahren umweltschonender, sicherer und komfortabler machen sollen. Gleichzeitig führen diese Entwicklungen damit zu einem immer höheren Anteil der Elektronik im Kraftfahrzeug. Das stellt die traditionell aus dem Maschinenbau stammende Automobilindustrie vor die große Herausforderung, das Kraftfahrzeug als elektronisches und zunehmend informationstechnisches System in seiner Entwicklung beherrschbar zu machen. Da viele der aktuellen Probleme auf dem Gebiet der Kraftfahrzeugelektronik ihren Ursprung in den gewachsenen Strukturen haben, wird im Folgenden ein Überblick über ihre geschichtliche Entwicklung gegeben. Daraus lassen sich Tendenzen für die Zukunft ableiten und damit auch die Anforderungen für eine zukünftige Elektronikarchitektur bestimmen.

2.2 Die Entwicklung der Kraftfahrzeugelektronik

2.2.1 Anfänge der Kraftfahrzeug-Elektrik

In der frühen Phase der Automobilgeschichte existierte faktisch noch keinerlei Kraftfahrzeugelektrik und das Gebiet der Elektronik gab es zu dem Zeitpunkt noch nicht[2]. „Angelassen“ wurde der Motor mit Handkurbel und Muskelkraft und bei Dunkelheit wurde einfach auf das Fahren verzichtet. Erst in den zwanziger Jahren, als immer mehr zivil genutzte Fahrzeuge auch nachts die Straßen nutzten, ergab sich die Notwendigkeit für eine Beleuchtung in den Fokus der Automobilbauer. Allerdings tat sich hier die Elektrik schwer, sich im Automobil durchzusetzen. Obwohl abgeschlossen in ihren grundsätzlichen Erfindungen, fehlte es der Elektrik noch an aus daran resultierenden einsatzfähigen Produkten. Somit blieben trotz Ausnahmen wie der schon ausgereiften Dynamomaschine, deren Entwicklung nicht zuletzt durch den 1. Weltkrieg vorangetrieben wurde, für einige Zeit Lampen mit Kerzen, Petroleum oder Acetylen die einzigen Beleuchtungsquellen für Automobile.

Erst langsam tauchten vereinzelt Fahrzeuge mit Dynamomaschine auf, die Licht aus Elektrizität erzeugten. 1915 begann Ford mit der Einführung des elektrischen Lichts. Hierbei stellte sich auch sofort die Frage nach Licht bei stehendem Motor bzw. Fahrzeug. Die Batterie als Energiespeicher war zwar schon verfügbar, die gesamte Beleuchtung musste aber in den automobilen Anfängen als „Sonderausstattung“ extra bezahlt werden. Die Preise für eine solche ständig verfügbare Beleuchtung waren beachtlich, so dass der am Reifen mitlaufende Dynamo und die von ihm gespeiste Lampe für lange Zeit die gesamte Kraftfahrzeugelektrik bildete.

Der einfache Dynamo hatte den weiteren Nachteil, dass er bei widrigen Straßen- und Wetterverhältnissen nur ungenügend funktionierte, weil die Kopplung zwischen Reifen und Dynamo behindert wurde. Somit führte an der Batterie als Energiespeicher kein Weg vorbei. Dabei entstand jedoch das Problem, diesen Speicher effizient zu bedienen, d.h. ihn immer gleichmäßig zu laden, aber andererseits vor Überladung zu schützen. Hinzu kam die Aufgabe bei ruhendem Fahrzeug elektrische

[2] Der Begriff „Elektrik“ umfasst als Überbegriff prinzipiell alle elektrischen und elektronischen Systeme. Innerhalb dieser Arbeit bezeichnet „Elektrik“ konkreter die elektrischen Systeme, die hauptsächlich der Versorgung dienen und vor allem Antriebe und Beleuchtungen umfassen. Dazu im Gegensatz stehen die elektronischen Systeme, die Signale verarbeiten bzw. messen und regeln.

Verbraucher wie Anlasser, Licht oder die Hupe durch die Batterie zu versorgen. Im Fahrbetrieb übernahm der Dynamo oder später auch die Lichtmaschine die Energieversorgung. Auftretende Leistungsspitzen mussten ebenfalls von der Batterie ausgeglichen werden. In den nächsten Jahrzehnten bestanden die Fortschritte in der Kraftfahrzeugelektrik deshalb zumeist in der Regelung zwischen Batterie und Lichtmaschine. Später kamen noch vereinzelt weitere elektrische Sonderaustattungen wie Scheibenwischer und Lüftung hinzu. Die elektrische Anlage eines Kraftfahrzeugs setzte sich in den 30er Jahren im Wesentlichen aus Zündanlage, Gleichstrom-Lichtmaschine mit Spannungsregler, Scheibenwischer, Signalhorn, Scheinwerfern und Leuchten zusammen. Diese Anlage blieb bis in die 50er Jahre weitgehend unverändert [Schwarz78] und der Anteil der Elektrik an den Entstehungskosten eines Automobils blieb über diese Jahre nahezu konstant auf einem sehr niedrigen Niveau.

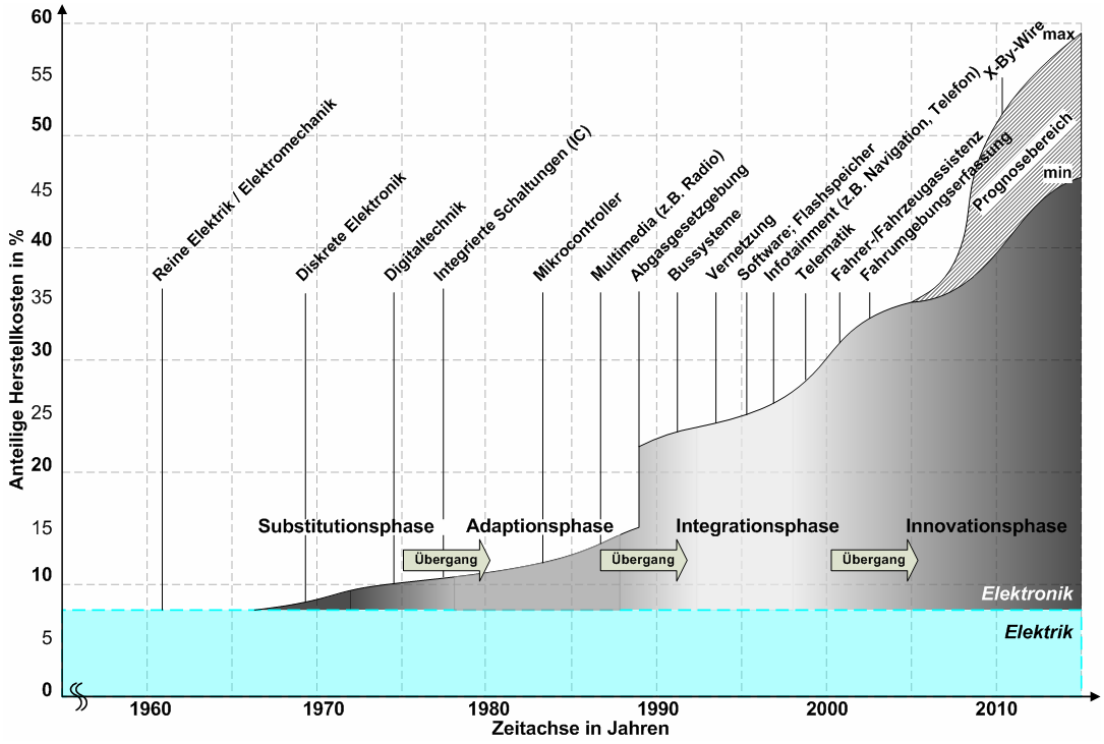

Abbildung 5: Evolutionsmodell der Kraftfahrzeugelektronik

Mit der Erfindung des Transistors im Jahre 1947 änderte sich dieser Stellenwert. Das Gebiet der Elektronik war entstanden und hielt in den späten sechziger Jahren auch langsam Einzug in das Automobil. Nun war es möglich, vorkommende Ströme mit sehr kleinen Bauelementen zu schalten. Es begann 1958 mit der Lichtmaschine als Gleichstromgenerator mit Variode, die 1962 dann von einem Drehstromgenerator mit Variode abgelöst wurde [Bosch02]. Seit dieser Zeit wächst der elektronische Anteil im Kraftfahrzeug ständig weiter. Die Entwicklung lässt sich in verschiedene Phasen unterteilen, die ineinander übergehen und zum Teil bis heute noch andauern. Einen Überblick darüber gibt die Abbildung 5. Hier werden der wachsende Anteil der Elektronikkosten in den einzelnen Phasen dargestellt und herausragende Meilensteine der Kraftfahrzeugelektronik zeitlich gekennzeichnet. Die Bezeichnung

der einzelnen Phasen sind an [TEhlers03] angelehnt und werden in den folgenden Abschnitten kurz vorgestellt.

2.2.2 Substitutions-Phase

In den Anfängen der Kraftfahrzeugelektronik bot sich mit der Erfindung des Transistors eine Möglichkeit, Ströme im Fahrzeug mit kleineren Bauelementen zu schalten. Aus diesem Grund wurden in der Folge große und schwere (elektro-)mechanische Steuerungen und Verstellsysteme durch elektronische Transistorschaltungen ersetzt. Diese zeichneten sich vor allen Dingen durch geringes Gewicht, kleinen Bauraum und Wartungsfreiheit aus, wobei diese Schaltungen noch aus einzelnen diskreten Bauelementen aufgebaut waren. So kam 1967 im Volkswagen 1600 TL die erste elektronisch gesteuerte Benzineinspritzung – die D-Jetronik – auf den Markt und ersetzte die bis dahin typischen mechanischen Einspritzsysteme [Leonhard01], [Internet1].

Ein anderes Beispiel aus dem Bereich der Komfortelektronik, die etwas günstigere Umweltbedingungen als im Motor vorfindet, sind die Blinkgeber, die sich besonders gut durch Elektronik ersetzen ließen. Die früheren Blinkgeber erzeugten das periodische Öffnen und Schließen der Kontakte in den Leitungen der Blinklampen durch einen mit Strom durchflossenen Hitzedraht in Verbindung mit einem Elektromagnet. Durch Betätigen des Blinkers erwärmt sich der Hitzedraht durch den Lampenstrom und öffnet durch die damit verbundene Längenänderung den vorgespannten Blinkanker. Damit wird der Lampenstromfluss unterbrochen, der Hitzedraht erkaltet und zieht sich zusammen, was den Blinkanker in seine Ruhestellung zieht. Dadurch sind die Kontakte wieder geschlossen, worauf erneut ein Strom fließen kann und die Prozedur sich wiederholt. Die Nachteile dieser Technik liegen auf der Hand. Die Blinkfrequenz ist nur ungenau einstellbar, die beteiligten Elemente sind Verschleiß und Alterung unterworfen sowie stark von äußeren Einflüssen (z.B. Bordnetzspannung) und Umweltbedingungen (z.B. Temperatur) abhängig. Durch elektronische Blinkgeber konnten diese Nachteile vermieden werden, indem durch Betätigen des Blinkers ein astabiler Multivibrator angestoßen wurde, der wiederum ein Relais ansteuert, das die Kontakte öffnet und schließt. Zusätzlich konnte damit der Ausfall einer Lampe durch die damit verbundene Frequenzerhöhung angezeigt werden. Später wurden diese elektronischen Komponenten durch integrierte Schaltungen ersetzt und um „digitale Intelligenz“ erweitert, so dass Zusatzfunktionen wie z.B. Warnblinken, Anhängerblinklicht usw. mit einer einzigen Schaltung möglich wurden. Weitere Beispiele sind die Scheibenwischanlagen, Alarmanlagen oder Anzeigesysteme, wie in [Schwarz78] beschrieben.

2.2.3 Adaptions-Phase

In der Substitutionsphase, in der mechanische Komponenten durch elektronische ersetzt wurden, konnte die Elektronik beweisen, dass sie die an sie gestellten Anforderungen auch im rauen Umfeld eines Kraftfahrzeugs erfüllt. Durch den Übergang von der Analogtechnik zur Digitaltechnik mit ihren Technologien wie beispielsweise CMOS oder TTL wurde die Umstellung von zeitkontinuierlichen, analogen Signalen auf zeitdiskrete digitale (Regel-)Signale möglich. Damit war man

in der Lage, komplexere Abläufe zu koordinieren und die elektronischen Systeme besser auf die Anforderungen im Kraftfahrzeug abzustimmen. So konnte z.B. die elektronische Zündung feiner an den konstruktiv unveränderten Verbrennungsmotor adaptiert und damit bessere Leistungsdaten und später auch verbesserte Verbrauchs- und Emissionswerte erzielt werden.

Mit dem Einsatz diskreter Bauelemente, die erst nach und nach durch integrierte Schaltungen (ICs) abgelöst wurden, waren der Komplexität der Schaltungen jedoch Grenzen gesetzt. Außerdem fand die Mikroelektronik im Kraftfahrzeug denkbar ungünstige Umweltbedingungen vor, wie Temperaturbereiche zwischen –40°C bis +170°C, Vibrationen mit hohen Frequenzen, Feuchtigkeit usw. [Förster84]. Der Anteil der Elektronik an den Herstellungskosten eines Kraftfahrzeugs blieb deshalb auch zu Beginn der Adaptionsphase relativ gering. Die Erwartungen an die Elektronik waren hoch, ebenso aber die Sorgen um die Zuverlässigkeit. So war [Förster84] im Jahr 1984 der Auffassung, „... ein nachhinkendes evolutionäres Eindringen der Elektronik in die Kraftfahrzeugtechnik zu erleben, was für die Autobenutzer nicht unbedingt ein Nachteil zu sein braucht“.

Erst mit der Einführung des Mikrocontrollers und der zunehmenden Miniaturisierung der Elektronik nahm deren Anteil im Kraftfahrzeug weiter zu. Seitdem geht die Entwicklung der Kraftfahrzeugelektronik einher mit der Entwicklung der Technologien auf dem Halbleitersektor, wenn sie auch nicht ganz deren Innovationsgeschwindigkeit folgen kann (siehe Abschnitt 2.3). Es wurden erstmals Funktionen realisiert, die mit mechanischen Mitteln nicht umsetzbar waren, wie z.B. die Leerlaufdrehzahlregelung und die Lambdaregelung [Rech04].

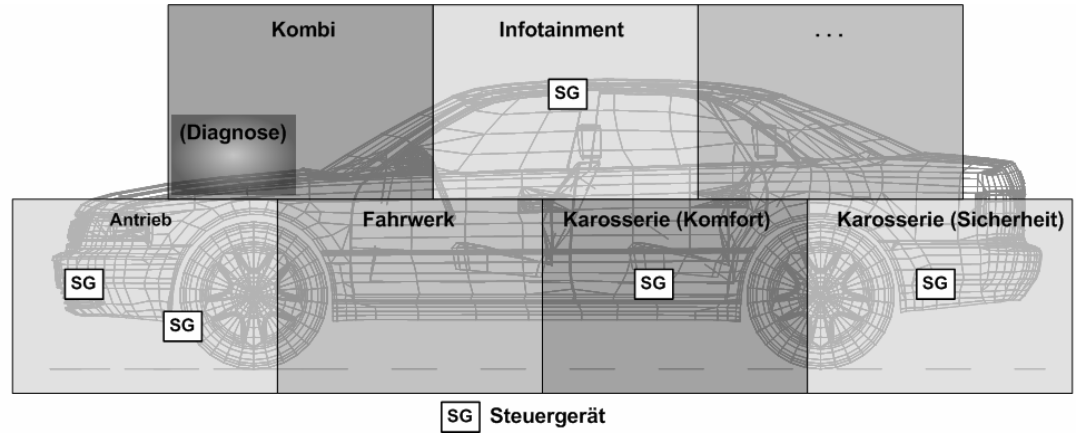

Abbildung 6: Funktionsdomänen im Kfz zum Ende der Adaptionsphase

Die elektronischen Steuergeräte, die im Fahrzeug in dieser Zeit eingesetzt wurden, arbeiteten weitgehend autonom und ohne gegenseitige Wechselwirkungen. Die Funktionen konnten deshalb eindeutig einem bestimmtem Steuergerät und seine Aufgabe in einem bestimmten Fahrzeugsubsystem wie Antriebsstrang, Fahrwerk oder Karosserie zugeordnet werden (siehe Abbildung 6). So können beispielsweise Motor- und Getriebesteuerung als klassische Systeme für den Antriebsstrang bezeichnet werden, während z.B. das Anti-Blockier-System (ABS) dem Fahrwerk oder Sitz- und Spiegelverstellung dem Komfortbereich des Karosseriesystems zugeordnet werden [Schäuffele03].

2.2.4 Integrations-Phase

Die dritte Evolutionsphase der Kraftfahrzeugelektronik kann als Integrationsphase bezeichnet werden, in der elektronische Komponenten in ein Fahrzeugelektronik-Gesamtsystem eingebunden sind. Sie wurde hauptsächlich durch die fortschreitende Entwicklung des Mikrocontrollers und die Vernetzung von Steuergeräten vorangetrieben.

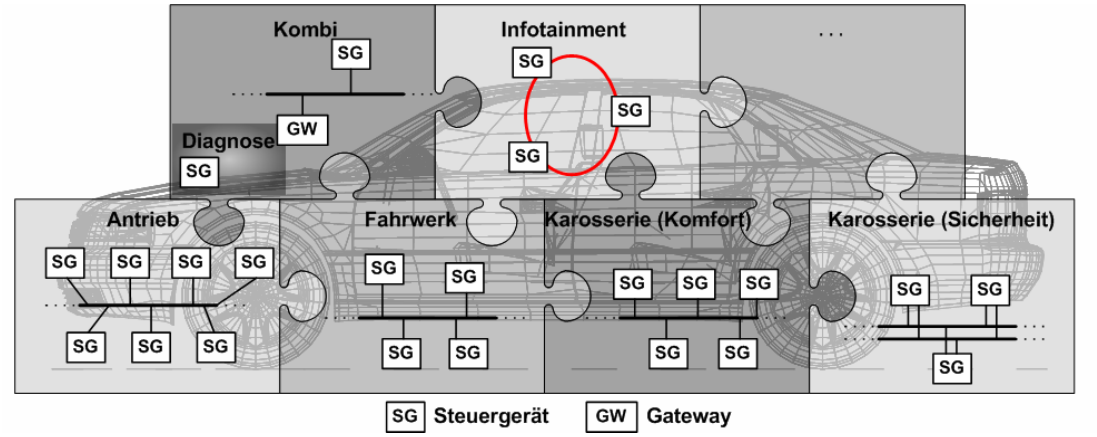

Abbildung 7: Vernetzte Funktionsdomänen in der Integrationsphase

2.2.4.1 Mikrocontroller

Die Integration des Mikrocontrollers in das Kraftfahrzeug musste allerdings wie jede Innovation erst Hindernisse überwinden. Durch ihre damals geringen Stückzahlen waren Mikrocontroller extrem teuer und damit den üblichen kundenspezifisch maßgeschneiderten Schaltungen auf den ersten Blick technisch und kostenmäßig unterlegen. Für den jeweiligen Kunden entworfene Schaltkreise konnten anhand der gestellten Anforderungen nahezu beliebig erweitert und die Prozessverarbeitungszeit durch den Aufbau von Parallelstrukturen reduziert werden. Mikrocontroller besitzen dagegen ein vorgegebenes Design mit einem festen Befehlsvorrat, der sequentiell abgearbeitet wurde. Der entscheidende Vorteil der Mikrocontroller ist ihre Flexibilität durch Software, mit der das Systemverhalten sehr schnell geändert werden kann. Dazu kamen durch die großen Fortschritte der Halbleitertechnik leistungsfähigere Mikrocontroller in immer kürzeren Entwicklungszyklen auf den Markt. Diese Umstände verhalfen dem Mikrocontroller zum Durchbruch im Kraftfahrzeug.

Bei der Entwicklung neuer Systeme wurde begonnen, die Funktionen auf mechanische und elektrische/elektronische Teilsysteme zu verteilen, um optimale Ergebnisse zu erzielen. Dieses Vorgehen lässt sich am Beispiel der elektronischen Motorsteuerung verdeutlichen. Durch die damit verbesserte Regelung der Kraftstoffzufuhr wurde eine signifikant gesteigerte Kraftstoffverbrennung im Zylinder erreicht, was niedrigere Emissionen bei höherer Motorleistung ermöglicht [TEhlers03]. Die nächste Stufe war dann die Verteilung der Funktionen auf verschiedene Steuergeräte, was erst durch Vernetzung und den Einsatz von Software möglich wurde.

2.2.4.2 Software

Die Software hielt zusammen mit der Einführung des frei programmierbaren Mikrocontrollers Einzug ins Fahrzeug, was sich am Beispiel des ABS belegen lässt. In Deutschland wurden ab 1978 Antiblockiersysteme produziert – zunächst auf Basis der damals üblichen kundenspezifisch maßgeschneiderten elektronischen Schaltungen. Obwohl der Sicherheitsgewinn von ABS schnell erkannt wurde, standen der erfolgreichen Verbreitung die Nachteile der frühen elektronischen Systeme im Weg – zu teuer, zu kompliziert und die Adaption auf unterschiedliche Autotypen dauerte zu lang. Durch den Einsatz von Mikrocontrollern konnten diese Nachteile behoben werden, indem die fahrzeugtyp- und herstellerspezifischen Eigenschaften des ABS in Software umgesetzt wurden. Damit wurde eine Serienfertigung von mikrocontroller- und softwarebasierten Systemen möglich, die heute Bestandteil aller elektronischen Steuergeräte im Fahrzeug sind. Im Fall des ABS kam 1984 das erste mikrocontrollerbasierte System in den USA in einem Ford Lincoln Continental auf den Markt [Fennel05].

Anfangs spielte die Software eine untergeordnete Rolle bei der Entwicklung neuer Fahrzeuge. Sie war meist für ein bestimmtes Steuergerät konzipiert und für eine überschaubare Anzahl von Funktionen verantwortlich. Für den Automobilhersteller wurde dabei noch nicht zwischen Hard- und Software getrennt. Es wurde eben ein Steuergerät, bestehend aus Hard- und Software, vom Zulieferer eingekauft. Dies stellte insofern noch kein Problem dar, da die Software, nachdem sie programmiert und getestet war, mittels Masken-ROM (siehe Kapitel 3.2.3) fest und unveränderlich in das Steuergerät eingebaut wurde. Falls Softwarefehler vorhanden waren und später entdeckt wurden, mussten sehr kostenintensiv neue Masken-ROMs hergestellt und die kompletten Steuergeräte ausgetauscht werden. Allerdings waren Fehler in der Software auf Grund des geringen Code-Umfangs und der noch überschaubaren Testfälle relativ selten.

2.2.4.3 Vernetzung

Während zu Beginn der Integrationsphase die verschiedenen elektronischen Steuergeräte noch autonom arbeiteten, wurde ab Mitte der 80er Jahre mit deren Vernetzung innerhalb der einzelnen Subsysteme begonnen. Mittels leistungsfähiger Datenbussysteme, z.B. dem Controller Area Network-Datenbus (CAN-Datenbus, Kapitel 3.3.5) waren neue, übergeordnete (Software-)Funktionen möglich. Dieser Ansatz wird entsprechend seines jeweiligen Subsystems auch als integriertes Antriebsstrang-, integriertes Fahrwerks- oder integriertes Karosseriemanagement bezeichnet, bei dem einzelne Funktionen oft nicht mehr einem einzigen Steuergerät zugeordnet werden können. Durch die Vernetzung sind (Software-)Funktionen auch keinem bestimmten Subsystem mehr zuzuordnen. Beispielsweise ist die Antriebsschlupfregelung eine antriebsstrang- und fahrwerksübergreifende Funktion, da sie die Raddrehzahlen im Fahrwerk-Subsystem überwacht und gleichzeitig auf das Motormoment im Antriebsstrang wirkt.

2.2.4.4 Komplexität

Spätestens während der Integrationsphase war die Elektronik aus dem Automobil nicht mehr wegzudenken, was sich positiv bei Sicherheit, Emissionen, Verbrauch und Komfort auswirkte. Allerdings zeigten sich auch erste Schwierigkeiten im Einsatz der Elektronik. Durch die zunehmende Zahl mechanisch anfälliger Steckkontakte und das erhöhte Risiko von unerwarteten Seiteneffekten durch die hohe Vernetzungskomplexität resultierten Probleme mit der Zuverlässigkeit und Verfügbarkeit. So führte beispielsweise der inzwischen deutlich gesteigerte Verkabelungsaufwand zu erheblichen Einbußen an Qualität und Zunahme an Gewicht.

Man begann bereits zu diesem Zeitpunkt über Diagnosemöglichkeiten der, nach außen als Black-Box wirkenden, elektronischen Steuergeräte nachzudenken und diese in einfacher Form umzusetzen. Während die Vernetzung der Steuergeräte half, den Verkabelungsaufwand zu reduzieren, erhöhte sich im Gegenzug die Komplexität elektrischer Aspekte wie Busruhe oder definierte Signalpegel. Dazu traten neue Herausforderungen im Bereich des Datenmanagements und der Schnittstellen auf. Der Automobilbauer musste gewährleisten, dass ein Steuergerät vom Zulieferer A mit einem anderen vom Zulieferer B problemlos zusammen arbeitete und die ausgetauschten Daten konsistent waren. Gerade diese Problematik wurde in den Anfängen der Steuergerätevernetzung unterschätzt und ist bis heute eine der großen Herausforderungen der Automobilindustrie.

2.2.5 Innovations-Phase

Mittlerweile befinden wir uns im Übergang zur nächsten Phase, die als Innovationsphase bezeichnet werden kann. Diese zeichnet sich wesentlich dadurch aus, dass Erfindungen und Technologien aus anderen Bereichen wie Mikroelektronik und Kommunikationstechnik immer mehr zu innovativen Lösungen im Kraftfahrzeug beitragen, die mittlerweile weit über dessen „Grundfunktionen“ hinausgehen. Die Kraftfahrzeugelektronik wird dabei nicht mehr als ein Teilaspekt der verschiedenen Subsysteme gesehen, sondern als ein Fahrzeug-Gesamtsystem verstanden. Die Kommunikation bleibt hier nicht mehr allein auf das Fahrzeug beschränkt. Sie wird vielmehr über die Grenzen des Fahrzeugs hinaus ausgedehnt, um das Fahrzeug in seine Verkehrsumgebung einzubeziehen.

Die Weiterentwicklung der Fahrzeugelektronik bei den Automobilherstellern geht deshalb in Richtung der Beherrschung dieser Vernetzung und des Prozessmanagements hinsichtlich der Entwicklung neuer Funktionen. Dies führt zu einem Abwägen zwischen Innovationsnutzen, Aufwand und Kosten, denn nicht alles, was technisch machbar ist, muss auch sinnvoll sein.

Die Innovationsphase wird die zukünftige Entwicklung im Automobil prägen und wird daher detaillierter dargestellt. Im Anschluss daran, in den Abschnitten 2.3 und 2.4, werden die verschiedenen Aspekte der Elektronikentwicklung vorgestellt. Eine der wesentlichen Triebkräfte sind die Innovationen, nach denen diese Phase der Elektronikentwicklung benannt ist. Der Begriff „Innovation“ wird daher ausführlicher erklärt.

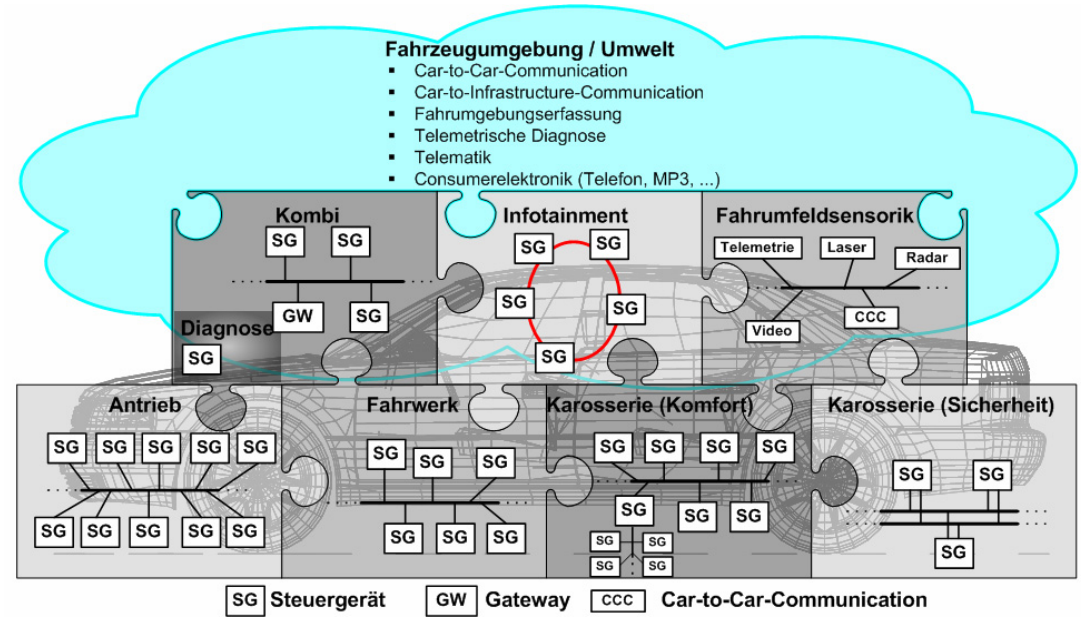

Abbildung 8: Stark vernetzte Funktionsdomänen in der Innovationsphase

2.2.5.1 Innovation

Das Wort „Innovation“ kommt aus dem Lateinischen (*novus* = neu, *innovatio* = etwas neu Geschaffenes) und wird im technischen Zusammenhang mit „neu Entwickeltem“ gebraucht, meist auch, um auf eine besondere Kreativität hinzuweisen. Es steht dabei als Synonym für neuen Nutzen oder neue Einsatzgebiete bereits existierender Produkte, Ideen, Konzepte oder Technologien. Etwas definierter ausgedrückt ist Innovation die Durchsetzung einer technischen oder organisatorischen Neuerung und nicht allein ihre Erfindung [Schumpeter12]. Andere Quellen wie [Rogers95] bezeichnen Innovation auch etwas allgemeiner als eine Idee oder ein Objekt, was von den Übernehmern (Adoptern) als neu angesehen wird.

Schon vor dieser Phase war die Automobilindustrie innovativ, doch mit der heutigen Leistungsfähigkeit der Elektroniksysteme wird dies zu einem beherrschenden Thema. Die Neuerungen gehen immer mehr über die eigentliche Grundfunktion des Kraftfahrzeugs, dem „Fahren“, hinaus. Sie haben ihren Ursprung dabei immer öfter in anderen Bereichen, insbesondere der Kommunikations- und Informationstechnik, und werden in das Fahrzeug adaptiert und integriert. Als Beispiel kann das „mobile Büro“ dienen. Verursacht durch das Verlangen unserer Gesellschaft nach immer mehr Mobilität (siehe Abschnitt 2.4) verbringen wir immer mehr Zeit in unseren Fahrzeugen. Es ist daher nahe liegend, andere Bereiche unseres Lebens in das Fahrzeug zu integrieren, wie beispielsweise Unterhaltungselektronik, Telefon oder Personal Digital Assistent (PDA). Einen guten Ausblick auf zukünftige Innovationen durch Elektronik im Automobilbereich geben [Schwab00] und [Winterkorn05].

2.2.5.2 Mikrocontroller

Die hohe Innovationsgeschwindigkeit der Halbleiterindustrie, die immer mehr Rechenleistung bei geringeren oder zumindest konstant bleibenden Kosten zur Verfügung stellt, begünstigte die Verbreitung des Mikrocontrollers im Kraftfahrzeug,

wie in Abschnitt 2.3 dargestellt wird. Die Anzahl elektronischer Steuergeräte, die mittlerweile schon mehrere Mikrocontroller enthalten können, wuchs in den letzten Jahren beständig weiter, was Abbildung 9 verdeutlicht. Besonders der Bereich „Infotainment“ im Kraftfahrzeug verlangte durch neue Technologien auf Gebieten der Telematik (Navigation), der Kommunikation (Telefon) und der Unterhaltungselektronik (DVD, DAB) nach immer höherer Rechenleistung.

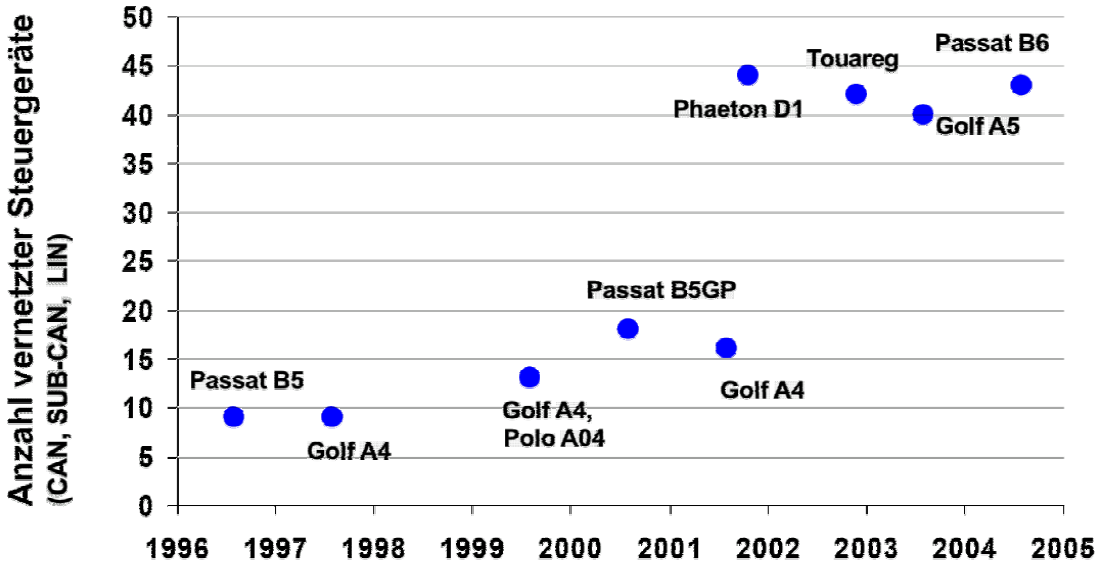

Abbildung 9: Entwicklung der Steuergeräteanzahl über die Fahrzeuggenerationen am Beispiel des Volkswagen-Konzerns [Neumann04]

So finden sich im Kraftfahrzeug heute eine Vielzahl unterschiedlicher spezialisierter Mikrocontroller wieder. Die anfänglich verbauten 4-Bit Mikrocontroller wurden sehr schnell durch 8-Bit Mikrocontroller abgelöst, die heute nur noch für einige einfache Aufgaben verwendet werden. Für anspruchsvollere Einsatzgebiete, die hohe Rechenleistung verlangen, werden zum größten Teil 16-Bit Mikrocontroller eingesetzt. Mittlerweile werden für fortschrittliche Embedded-Systeme wie Motor- oder Getriebesteuerungen teilweise schon 32-Bit Mikrocontroller benutzt. Diese müssen Echtzeit-Anforderungen genügen, da sonst eine exakte Regelung des Verbrennungsmotors nicht gewährleistet ist. Die Architektur solcher Prozessoren unterscheidet sich von solchen, die z.B. für den PC entwickelt werden [Hoika02]. Diese können dagegen im Infotainmentbereich eingesetzt werden, der nicht ganz den harten Einsatzbedingungen einer Motorsteuerung genügen muss. Die fortschreitende Integrationsdichte ermöglicht auch die Einbindung von Funktionen zur digitalen Signalverarbeitung (DSP) auf Mikrocontrollern. Somit lassen sich Funktionen in Software realisieren, die vorher in Hardware implementiert waren.

Mit zunehmender Anzahl von leistungsfähigen Mikrocontrollern im Kraftfahrzeug wächst auch der Softwareanteil sehr stark. Es gibt verschiedene Möglichkeiten, wie ein Mikrocontroller seine Software speichern kann. Für die Art des Programmspeichers lässt sich derzeit eine ähnliche Entwicklung beobachten, wie sie während der Integrationsphase zur Ablösung der kundenspezifischen Schaltungen durch Mikrocontroller führte. Heute steht die Automobilindustrie auf Grund des gestiegenen Softwareanteils vor der Entscheidung, die für sie günstigeren erprobten

Masken-ROMs gegen teurere, aber flexible Flash-ROMs (siehe Kapitel 3.2.3) abzulösen. Sie erlauben eine nachträgliche Änderung der Software und damit des Systemverhaltens.

2.2.5.3 Software

Die Software gilt heute als einer der größten Innovationstreiber der Kraftfahrzeugelektronik. Man schätzt, dass ca. 70% bis 80% der Innovationen im Kraftfahrzeug durch Software getrieben sind und erwartet eine Verdoppelung der Codegröße alle 2-3 Jahre [Schrey02]. Dies liegt hauptsächlich an der hohen Flexibilität, Wiederverwendbarkeit und den auf den ersten Blick relativ geringen Kosten auf Grund der fast kostenneutralen Vervielfältigungsmöglichkeit. Allerdings unterschätzte die Automobilindustrie bislang den hohen methodischen Aufwand der Softwareentwicklung. Anforderungsanalysen und Regelungen wie beispielsweise für die Hardwareentwicklung sind kaum vorhanden. Historisch bedingt erfolgt ein großer Teil der Softwareentwicklung nicht geplant oder unstrukturiert. Dazu kommt, dass sich Software im Grunde jederzeit ändern lässt. Dieser Entwicklungszyklus muss mit denen der Mechanik- und Elektronikkomponenten synchronisiert werden. Ein weiterer Unterschied zur klassischen Mechanik liegt in der absoluten Intoleranz von Software gegenüber Fehlern. Diese Eigenschaft ist verantwortlich dafür, dass die Software neben der Fahrzeugbatterie die häufigste Pannenursache moderner Kraftfahrzeuge darstellt [Kifmann02], [Kruse04].

Die Entwicklung fehlerfreier Software ist mit sehr viel Aufwand und Kosten verbunden. Daher wird man Fehler nie ganz ausschließen können. Durch die Möglichkeit der Aktualisierung (Update), die Software bietet, können auf diese Weise Fehler auch nachträglich korrigiert werden. Aus diesem Grund gehen die Fahrzeughersteller dazu über, einen Großteil der verbauten Steuergeräte updatefähig (flashbar) auszulegen. Dadurch, dass die Anzahl der verbauten Steuergeräte nicht mehr beliebig steigerungsfähig ist, der Funktionsumfang aber weiter zunimmt, wird die Software in den Steuergeräten immer umfangreicher und komplexer. Damit steigen deren Fehlerwahrscheinlichkeit und deren Umfang, der in ein Fahrzeug geflasht werden kann oder muss.

Die Automobilhersteller modularisieren die Software immer weiter, um unabhängiger von Hardware und Zulieferern zu werden. Mit Hilfe der Updateprogrammierung soll es zukünftig möglich sein, einzelne Module in den Steuergeräten per Flashen auszutauschen. Die Vision der Automobilhersteller geht dahin, die Software als Produkt verkaufen zu können. Diese wird als Zusatzausstattung angeboten und gegen Bezahlung per „Softwaretankstelle“ in das Fahrzeug geflasht. Zukünftige Kommunikationsarchitekturen müssen diese Szenarien berücksichtigen und unterstützen [Herrmann03], [Müller03].

2.2.5.4 Vernetzung

Die Vernetzung der einzelnen Subsysteme mit dem CAN-Datenbus, die in Anlehnung an die Informationstechnik mittlerweile als Domänen bezeichnet werden, ist in der Innovationsphase als DeFacto-Standard etabliert. Sie wird in den Bereichen, für die der CAN-Datenbus nicht konzipiert wurde, mittlerweile durch

besser angepasste Bussysteme ergänzt. Beispielsweise werden intelligente Sensoren und Aktoren, bei denen es um kleine Datenraten und geringe Kosten geht und bei denen der CAN-Datenbus zu teuer und zu aufwendig wäre, durch den Local Interconnect Network-Datenbus (LIN-Datenbus) vernetzt. Im Multimediabereich, der einen immer größeren Stellenwert in modernen Fahrzeugen einnimmt, benötigt man dagegen sehr große Datenraten mit etwas geringeren Anforderungen an die Datensicherheit und Verfügbarkeit. In diesem Bereich konnte sich in den letzten Jahren der Media-Oriented Systems Transport – Datenbus (MOST-Datenbus) etablieren, der eine optische Übertragungstechnik mittels Lichtwellenleiter nutzt. Einen Überblick über den Stand heutiger und zukünftiger automobiler Datenbusse gibt Kapitel 3.3.

Heutige Kraftfahrzeuge sind bezüglich ihrer Vernetzung so ausgelegt, dass sie im Falle eines Fehlers im Kommunikationsnetzwerk in einen sicheren Zustand gelangen. Dies kann entweder den Notlauf[3] bedeuten oder aber, wahrscheinlicher, ein stehendes Fahrzeug. Dazu ist ein mechanischer Durchgriff auf die zur Steuerung des Fahrzeugs relevanten Systemen wie Bremsen und Lenkung gesetzlich vorgeschrieben. Dieser gewährleistet bei Ausfall der Elektronik das Übergehen in einen sicheren Zustand ohne Gefahr für Insassen und Umwelt. Mit zunehmender Leistungsfähigkeit elektronischer Komponenten und Kommunikationssysteme wächst der Wunsch, diesen mechanischen Durchgriff durch elektrische Kommunikationsverbindungen zu ersetzen, was als „X-by-Wire“ bezeichnet wird. Das „X“ steht dabei für verschiedene Anwendungen wie Bremsen („Brake-by-Wire“) oder Lenkung („Steer-by-Wire“). Diese Systeme versprechen höhere Zuverlässigkeit, bessere Integration in das elektronische Gesamtsystem und ermöglichen darüber hinaus neue Funktionen bis hin zum automatisierten Führen des Kraftfahrzeugs. Weiterhin verringern sie durch den Verzicht auf mechanische Teile das Gewicht, vereinfachen die Wartung und reduzieren damit die Kosten eines Fahrzeugs. Grundlage solcher X-by-Wire-Systeme ist eine fehlersichere und deterministische Vernetzung mit hoher Datenübertragungsrate, die mit den herkömmlichen Datenbussen wie CAN nicht realisiert werden kann. Dafür müssen neuartige Bussysteme wie FlexRay oder TTP (siehe Kapitel 3.3.8), die derartige Anforderungen erfüllen, in das Fahrzeug integriert werden. Dies erfordert neue Kommunikationsarchitekturen im Kraftfahrzeug und neue Entwicklungsprozesse in der Automobilindustrie [Seiffert05], da somit das Kommunikationsnetzwerk ein sicherheitskritisches System darstellt.

Auf der anderen Seite wird eine immer weitere Öffnung dieses Kommunikationsnetzwerks für Kundenwünsche verlangt. Im Bereich Infotainment wird mehr und mehr die Integration von Unterhaltungs- und Kommunikationselektronik gefordert, die bei weitem nicht die Sicherheitsstandards der Automobilindustrie erfüllen. So kommunizieren schon heute in Oberklassefahrzeugen Mobiltelefone und MP3-Player[4] mit dem Fahrzeug-Datennetz, das im Oberklassesegment den Fahrzeuginsassen schon Internetzzugang ermöglicht

[3] Notlauf heißt, dass die Elektronik einen Fehler diagnostiziert hat und dafür Sorge trägt, dass das Fahrzeug sich nur noch innerhalb sicherer Parameter bewegen lässt. Dies könnte beispielsweise eine Drehzahlbeschränkung zur Folge haben.

[4] MP3-Player – (tragbare) Geräte zur Musikwiedergabe, hauptsächlich für das MP3-Format, ein standardisiertes Musikkompressionsverfahren

[Specks01]. Die Automobilhersteller müssen über geeignete Maßnahmen wie Architekturauslegung oder Schnittstellenfestlegungen sicherstellen, dass diese Kundenwünsche erfüllt werden und gleichzeitig die Sicherheit und Systemintegrität des Fahrzeugs gewährleistet sind.

Eine der zukünftigen Innovationen, die derzeit diskutiert werden, ist die Fahrzeug-Fahrzeug-Kommunikation (Car-to-Car Communication – CCC) bzw. Fahrzeug-Infrastruktur-Kommunikation. Dabei bilden Fahrzeuge während der Fahrt untereinander Ad-hoc-Netzwerke und tauschen Informationen aus, die zur Kollisionsvermeidung, Verkehrsbeeinflussung oder auch zur Mauterfassung genutzt werden können. Weiterführende Details beschreiben [Franz04], [Lübke04] und [Specks05]. Bei der Fahrzeug-Fahrzeug-Kommunikation sind die Auswirkungen auf die Vernetzung des Kraftfahrzeugs noch nicht absehbar. Dabei sind insbesondere die Aspekte der Sicherheit, des autorisierten Zugriffs und nicht zuletzt der Datenschutz zu beachten. Klar ist jedoch, dass sich die Vernetzung über die Fahrzeuggrenzen hinweg ausdehnen wird und die Komplexität dadurch ansteigt. Falls sich diese Kommunikation vom internen Fahrzeug-Kommunikationssystem trennen ließe, würde man die Fahrzeug-Fahrzeug-Kommunikation nicht mehr zum Kommunikationssystem des Fahrzeugs zählen, sondern sie dem Bereich der Umfeldsensorik zuordnen.

2.2.5.5 Komplexität

Der vermehrte Einsatz der Elektronik ab der Integrationsphase vergrößerte die damit verbundenen Anforderungen für die Bereiche Entwicklung, Qualität, Verfügbarkeit und Wartung. Für die traditionell aus dem Maschinenbau stammenden Automobilbauer stellte die Elektronik ein völlig neues Gebiet dar, das zudem immer mehr von der Informationstechnik abgelöst wird. Wurden anfänglich für neue Funktionen zusätzliche Steuergeräte an den Datenbus angeschlossen, so reicht dieser Ansatz schon lange nicht mehr aus. Die starke Vernetzung der Steuergeräte miteinander erfordert einen ganzheitlichen Ansatz einer System- und Kommunikationsarchitektur. Funktionen lassen sich nicht mehr einem einzelnen Steuergerät zuordnen. Diese werden zukünftig Software von mehreren Zulieferern enthalten. Zeitgesteuerte Datenbusse stellen hohe Anforderungen an Ressourcen und Architekturen, die Umfeldsensorik verlangt hohe Datenraten und eine genaue Synchronie, die Systemintegrität des Fahrzeugs muss auch nach Softwareupdates gewahrt bleiben, um nur einige Schwerpunkte der Komplexität in zukünftigen Fahrzeugen zu nennen. Mit der Einführung der X-by-Wire-Technologie übernimmt die Elektronik im Kraftfahrzeug mehr und mehr sicherheitskritische Aufgaben. Die daraus resultierenden zusätzlichen Anforderungen an Echtzeitfähigkeit sowie Fehler- und Störsicherheit steigern die Komplexität weiter. Durch die Vernetzung des Fahrzeugs mit seiner Umwelt wird diese nochmals weiter erhöht. Die Automobilindustrie versucht das durch verschiedene Initiativen und Bestrebungen beherrschbar zu machen, von denen die wichtigsten im Abschnitt 2.5 kurz vorgestellt werden.

2.3 Technische Triebkräfte der Kraftfahrzeugelektronik

Als die wichtigste Triebkraft der Kraftfahrzeugelektronik kann seit Erfindung des Transistors im Jahre 1947 die Halbleiterindustrie gelten. Sie ermöglichte den Mikrocontroller, durch den erst intelligente elektronische Systeme nach heutigem Verständnis realisierbar wurden. Die technologischen Fortschritte der Halbleiterbranche erlauben eine immer höhere Leistungsfähigkeit der Mikroprozessoren. Möglich wird das durch eine ständig wachsende Integrationsdichte von Halbleiterelementen auf einem Chip, die auch heute noch der von Gordon E. Moore 1965 postulierten Kurve folgt, die im langfristigen Durchschnitt alle 18 Monate eine Verdoppelung der Integrationsdichte fordert. In Abbildung 10 ist diese Kurve für Prozessoren aus dem PC-Consumer-Bereich, von dessen Fortschritten die Automobilelektronik profitiert, dargestellt. Gekoppelt mit den weiter steigenden Integrationsdichten sind höhere Rechenleistungen und ein zunehmender Speicherausbau bei den Mikrocontrollern.

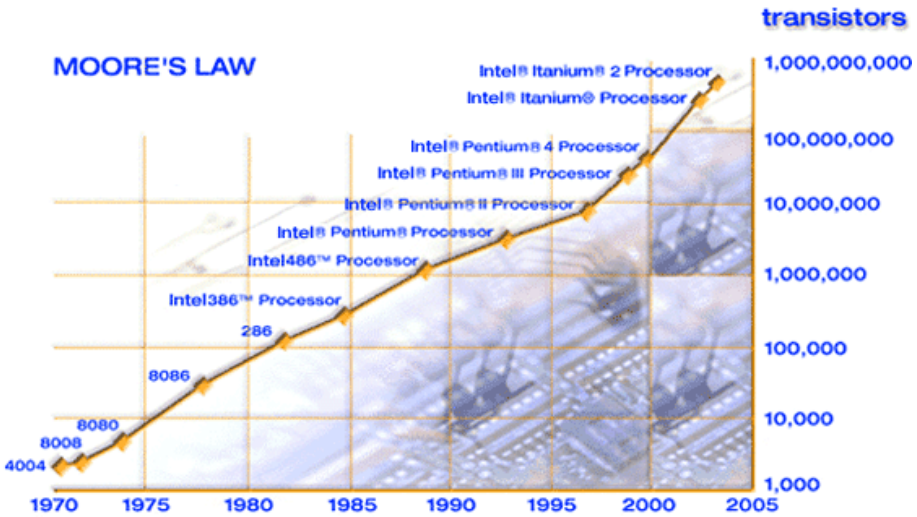

Abbildung 10: Das Gesetz von Gordon E. Moore [Intel05]

Die Mikroprozessoren für den Embedded-Bereich, die im Kraftfahrzeug eingesetzt werden, bleiben mit ihren Integrationsdichten und den damit verbundenen Verarbeitungsgeschwindigkeiten zwei bis drei Jahre hinter den führenden Technologien des PC-Consumer-Bereichs zurück. Dies liegt an den höheren Anforderungen, die im Automobilbereich an Elektronik und Prozessoren gestellt werden. Dort sind MIPS/mA[5] (Leistungsfähigkeit pro Stromaufnahme) wichtiger als Taktraten in MHz. Hinzu kommen weitere Anforderungen wie der erweiterte Temperaturbereich, die längere Ersatzteilverfügbarkeit, die Integration von Flashspeicher, die Einbindung analoger Komponenten in die Embedded-Technologie und nicht zuletzt ist die EMV ein sehr wichtiger Zielparameter im Design [Teepe00]. Damit entsteht eine Leistungslücke zwischen den schnellen Technologien des PC-Consumer-Bereichs und den robusten, aber viel langsameren Rechnern der Automobilelektronik.

[5] MIPS – engl. Million Instructions Per Seconds / mA - Milliampere

Bei einem Vergleich der historischen Entwicklungen der Computerindustrie mit der Automobilelektronik wird deutlich, dass der Schlüsselfaktor in den Bereichen Hard- und Software die Schaffung von Standards waren und sind [Teepe00]. Die Automobilindustrie versucht aus diesem Grund, ebenfalls Standards zu etablieren, um mit der Innovationsgeschwindigkeit der Halbleiter- und Computerindustrie Schritt zu halten. Einige dieser Bemühungen stellt der Abschnitt 2.5 vor.

Durch den zunehmenden Einsatz und die wachsende Leistungsfähigkeit von Mikrocontrollern entstehen immer mehr verteilte elektronische Systeme im Fahrzeug, verstärkt durch die wachsende Anzahl an intelligenten Aktoren und Sensoren. Für diese verteilten elektronischen Systeme ist der Bereich der Kommunikation von zentraler Bedeutung, der sich in die Industrie- und Feldbusse, das Ethernet und Internet sowie die drahtlosen Netzwerktechniken unterteilen lässt. Der Bereich der Datenbusse wird im Kapitel 3.3 ausführlich behandelt, drahtlose Techniken im Kapitel 3.4 vorgestellt. Diese Kommunikationstechnologien profitieren ebenfalls sehr stark von den Fortschritten der Halbleiter- und Mikroprozessorentechnik.

Gleichzeitig besitzt die Kommunikationsbranche selbst eine hohe Innovationsgeschwindigkeit, wie tragbare Telefone, PDAs oder MP3-Player zeigen. Die Automobilindustrie kann auf diese Technologien zurückgreifen und für ihre Belange wie beispielsweise Car-to-Car Communication oder ähnliche Innovationen einsetzen. Zugleich stellen die hohe Innovationsgeschwindigkeit und kurze Entwicklungszyklen die Automobilindustrie vor große Herausforderungen.

2.4 Automobilindustrie im Umbruch

Das Automobil ermöglicht uns die hohe individuelle Mobilität, die unsere Umwelt prägt und damit die Grundlage unserer heutigen Gesellschaft darstellt [ifmo02], [ifmo05], [Lukas04]. Mit der zunehmenden Verbreitung des Automobils und seiner intensiven Nutzung wachsen die damit verbundenen Probleme weiter an, da auf Grund der immer häufigeren Staus, den damit verbundenen Umweltbelastungen und durch den Bedarf an begrenzten fossilen Brennstoffen der Mobilität immer engere Grenzen gesetzt werden. Dazu greift der Grundmechanismus „Verkehr schafft Verkehr“ [Heinze79], bei dem die Mobilität ebenfalls weiter eingeschränkt wird und sich zu einem selbst hemmenden System entwickelt. Die Politik versucht dieser Entwicklung entgegenzusteuern, wobei die klassischen Lösungsansätze wie Verkehrswegeausbau und Reglementierung überdacht werden müssen [Petersen99]. Einen Beitrag dafür leistet die Automobilindustrie, in dem sie sich durch freiwillige Selbstverpflichtungen zur Sicherheitserhöhung und Emissionsverringerung verpflichtet. Der Zusammenhang zwischen Verkehr, Mobilität und Innovationen in Verbindung mit einem Ausblick auf die Zukunft der Verkehrstechnik wird umfassend bei [Herbst06] beschrieben.

Für die Automobilindustrie muss aber vor allen Dingen die Aufrechterhaltung der Mobilität oberste Priorität haben – es ist ihr wichtigstes Verkaufsargument. Wie [ifmo05] und [WBCSD04] zeigen, wurde dies von vielen Automobilherstellern erkannt und es wird an Lösungen gearbeitet. Da besonders in Deutschland die Verkehrswege weitestgehend ausgebaut sind, die Verkehrsdichte aber weiter steigt, müssen neue Fahrer- und Fahrzeugassistenzsysteme entwickelt werden, die eine

aktive Verkehrslenkung ermöglichen. Dafür sind Informationen, die online zwischen Fahrzeugen untereinander und mit der Infrastruktur (Car-to-X Communication) ausgetauscht werden, unverzichtbar, was den Kommunikationsbedarf der Fahrzeuge weiter steigern wird.

Erschwerend kommt für die Automobilindustrie der Wandel des Marktes vom Wachstums- zum Verdrängungsmarkt hinzu. Dies zeigt sich beispielsweise in der Markenkonzentration, die in den letzten Jahren stattgefunden hat. Beispiele sind Ford (u.a. Aston Martin, Jaguar, Land Rover, Mazda, Volvo), GM (u.a. Cadillac, Chevrolet, Daewoo, Opel, Saab), Mercedes-Benz (DaimlerChrysler), Volkswagen (u.a. Audi, Bentley, Bugatti, Lamborghini, Seat, Skoda). Die Automobilhersteller gehen dazu von wenigen Fahrzeugmodellen mit langen Produktlebenszyklen und hohen Stückzahlen über zu einer großen Modellvielfalt mit kürzeren Produktlebenszyklen und geringeren Stückzahlen, um Nischenmärkte besetzen zu können. Das Bestehen in diesem Verdrängungsmarkt erfordert eine Differenzierung von den Mitbewerbern vor allen Dingen über Preis und Funktionalität. Besonders letzteres wird zukünftig eine große Rolle spielen, da die Grundfunktion des Automobils, das Fahren, vorausgesetzt werden kann und dabei keine große Differenzierung mehr möglich ist. Bei bestimmten Modellkategorien unterschiedlicher Hersteller sind kaum noch signifikante Unterschiede bei Leistungs-, Fahr- und Komforteigenschaften auszumachen. Hier bietet sich die Software als Differenzierungsmerkmal an, bei dem nicht-kundenrelevante Funktionen von Standardsoftware bewältigt werden und spezielle Funktionen, die den Charakter des Fahrzeugs ausmachen, direkt in der Verantwortung des jeweiligen Automobilherstellers liegen [Broy98].

Hohe Anforderungen an zukünftige Kraftfahrzeuge werden über freiwillige Selbstverpflichtungen der Automobilindustrie wie z.B. Verringerung des CO_2-Ausstoßes um 25% bis 2008 [Kozlowski04] oder die Halbierung der Anzahl der Unfalltoten bis 2010 [RSAP] und über gesetzliche Randbedingungen wie das kalifornische Zero Emission Law oder die europäische Abgasnorm Euro5 gestellt. Diese können nur mit Hilfe neuer Lösungen auf der Grundlage elektronischer Systeme bewältigt werden. Trotz der damit steigenden Systemkomplexität muss gleichzeitig eine weiter erhöhte Qualität der Fahrzeuge bereitgestellt und der Kostenanstieg begrenzt werden, um wettbewerbsfähig zu bleiben. Der wesentliche Schlüssel zur Beherrschung der weiter steigenden Systemkomplexität liegt in den künftigen Systemarchitekturen [Reichart05].

2.5 Aktuelle Forschungs- und Standardisierungsinitiativen

Die in den vorherigen Abschnitten skizzierte Evolution der Kraftfahrzeugelektronik stellt die Automobilindustrie vor große Herausforderungen. Um auf diese vorbereitet zu sein, sind in den letzten Jahren verschiedene Initiativen gegründet worden, von denen die wichtigsten kurz vorgestellt werden.

2.5.1 Intelligenter Verkehr und nutzergerechte Technik (INVENT)

Die Initiative „INVENT“ ist ein Zusammenschluss von 23 Unternehmen aus Automobil-, Zuliefer-, Elektronik-, Telekommunikations- und IT-Industrie,

Logistikdienstleistern, Softwarehäusern sowie Forschungsinstituten [INVENT] und hat sich folgende Ziele gesteckt:

- Erhöhung der Verkehrssicherheit
- Optimierung des Verkehrsflusses
- Ausgleich zwischen individuellen und gesellschaftlichen Zielen bei der Verkehrsentwicklung
- Selbstorganisation des Verkehrs durch Informationen im Verkehrsnetzwerk
- Nutzergerechte Technik

INVENT adressiert weniger die Kraftfahrzeugelektronik als solche, sondern versucht vielmehr mit deren Hilfe Möglichkeiten zu finden, die das Autofahren umweltschonender, sicherer und komfortabler machen. Damit bietet INVENT einen guten Überblick über zukünftige Fahrer- und Fahrzeugassistenzsysteme, die wiederum spezielle Anforderungen an die Kraftfahrzeugelektronik und deren Kommunikationsarchitekturen stellen werden. Im April 2005 wurden die Ergebnisse von INVENT präsentiert und es wurde beschlossen, Teilfragen insbesondere zu Fahrer- und Fahrzeugassistenzsystemen in gesonderten Projektinitiativen weiter zu untersuchen.

2.5.2 Strukturierter Entwicklungsprozess für die Softwareentwicklung in der Automobilelektronik (STEP-X)

In diesem Gemeinschaftsprojekt der Volkswagen AG mit vier Instituten der TU Braunschweig wird untersucht, mit welchen Methoden, Vorgehensmodellen und Werkzeugen ein durchgängiger Entwicklungsprozess für Kfz-Steuergerätesoftware geschaffen werden kann. Anhand beispielhafter Anwendungen - wie z.B. einem Pkw-Komfortsystem – werden dabei die Eigenschaften der eingesetzten Methoden und Werkzeuge evaluiert. Zu den Projektschwerpunkten gehört die Integration der Beschreibungsmittel für verschiedenartige Systemkomponenten zu einem Digitalen Lastenheft, das die Grundlage für eine durchgängig werkzeugunterstützte Entwicklung bilden kann. Das Digitale Lastenheft soll dann eine eindeutige und vollständige Beschreibung der Systemkomponenten gestatten und die bisherigen verbalen Funktionsbeschreibungen weitestgehend durch eine standardisierte grafische Darstellung ersetzen. Damit wird eine Integration von Test und Diagnose schon in den frühen Entwicklungsphasen ermöglicht.

2.5.3 Herstellerinitiative Software (HIS)

Der in den Kapiteln zur Evolution der Kraftfahrzeugelektronik angesprochene Wandel von Hardware zu Software führte zu einem ständig wachsenden Softwareanteil im Kraftfahrzeug. Diese Software muss für jedes einzelne Steuergerät in aufwendigen Prozessen geschrieben, getestet und versioniert werden, wobei die eigentliche, für den Kunden „erlebbare“, Steuergerätefunktion nur einen Bruchteil des Softwaregesamtumfangs ausmacht. Die deutschen Automobilhersteller Audi, BMW, DaimlerChrysler, Porsche und Volkswagen gründeten aus diesem Grund im

Jahr 2000 die Herstellerinitiative Software [HIS], die es sich zur Aufgabe gemacht hat, die Aktivitäten der einzelnen Marken im Bereich der Softwareentwicklung in nicht wettbewerbsrelevanten Bereichen zu bündeln und daraus Standards abzuleiten [Lange01]. Die Felder, die dabei bearbeitet werden, sind die so genannte Standardsoftware, Softwaretests, Simulation und Vereinheitlichung der Prozessreifegradermittlung bei Entwicklern im Bereich des Softwareengineerings. Die Aktivitäten und Ergebnisse bei HIS fließen direkt in AUTOSAR ein, da alle HIS-Mitglieder gleichzeitig Kern-Mitglieder bei der später gegründeten Initiative AUTOSAR sind [Chodura04].

2.5.4 Automotive Open System Architecture (AUTOSAR)

Eine der wichtigsten Industriepartnerschaften zur Beherrschung der Elektronikkomplexität im Kraftfahrzeug ist die 2003 gegründete „AUTomotive Open System ARchitecture“ (AUTOSAR)-Initiative [AUTOSAR], der bislang 77 Mitglieder (Stand: Sept. 2005) aus dem Automobilbereich angehören. AUTOSAR möchte eine einheitliche und offene Elektrik/Elektronik-Architektur für Kraftfahrzeuge schaffen, die es erlaubt, Software-Module herstellerübergreifend wieder zu verwenden bzw. mehrfach zu nutzen. Dazu definiert die weltweite Initiative standardisierte Funktionsschnittstellen und Basis-Softwaremodule. Damit verspricht man sich die Freisetzung wertvoller Ressourcen zur Umsetzung wirklich innovativer Lösungen [Ranft03]. Zur Herstellerinitiative Software HIS, die eher kurzfristige Standardisierungen existierender Systeme im Fokus hat, grenzt sich AUTOSAR über die längerfristige Perspektive ab.

Bis 2008 möchte AUTOSAR eine einheitliche Middleware-Architektur etablieren, auf der grundlegende Systemfunktionen im Fahrzeug wie Motor- und Karosserieelektronik sowie Multimedia- und Telematiksysteme aufbauen. Neben den Kostenvorteilen für Hersteller und Zulieferer durch erhöhte Skalierbarkeit und Flexibilität verspricht sich das Konsortium vor allen Dingen stabilere IT-Systeme mit geringeren Ausfallquoten. Bei der AUTOSAR-Middleware wurde von Anfang an darauf geachtet, diese auf eine zeitgesteuerte Architektur (TTA – Time Triggered Architecture) hin kompatibel zu entwickeln, wie sie beispielsweise FlexRay (siehe Kapitel 3.3.8) verlangt.

2.5.5 Weitere Standardisierungsbemühungen

Neben den bereits in den vorangegangenen Abschnitten vorgestellten Initiativen existieren weitere Standardisierungsgremien im Bereich der Kraftfahrzeugelektronik, zu denen ein kurzer Überblick folgt. Ihre Vielzahl zeigt vor allem, dass es einem einzelnen Hersteller nicht mehr möglich ist, die hohe Komplexität von Elektroniksystemen moderner Kraftfahrzeuge mit vertretbarem Aufwand zu bewältigen.

- ASAM – Association for Standardisation of Automation- and Measuring Systems
  Ein Zusammenschluss von über 60 Firmen, deren Zielsetzung die Standardisierung von Datenmodellen, Schnittstellen- und Syntaxbeschreibungen für vielfältige Applikationen ist.

- Automotive SPICE
  Das SPICE-Projekt (Software Process Improvement and Capability dEtermination) möchte über die Verbesserung des Software-Entstehungsprozesses die Qualität der Software verbessern und hat dafür verschiedene Bewertungsmethoden und Vorgehensmodelle in der ISO 15504 standardisiert. Das Automotive SPICE ist eine spezielle Interessengruppe dieses Projektes, bestehend aus internationalen Fahrzeugherstellern, und untersucht diese Fragestellungen in Bezug auf die Softwareentwicklung im Automobilbereich.

- CARTRONIC
  Eine Initiative von BOSCH, an der Continental Automotive Systems, ZF-Lenksysteme und ZF-Sachs beteiligt sind. Ziel bei dieser Zusammenarbeit ist die Erarbeitung von standardisierten Schnittstellen für Chassis-Systeme. Dabei stand zuerst ein einheitliches Verständnis des elektronischen Bremsweges im Vordergrund. Es gibt bereits abgestimmte Schnittstellen für aktive Lenk- und Fahrwerksysteme, den Antrieb sowie Fahrerassistenzsysteme. Die CARTRONIC-Schnittstellenbeschreibung wurde von AUTOSAR übernommen und soll dort auch auf andere Domänen übertragen werden.

- EASIS – Electronic Architecture and System Engineering for Integrated Safety Systems
  Die Zielsetzung dieser Initiative, an der Fahrzeughersteller, Zulieferer und Tool-Hersteller beteiligt sind, ist die Entwicklung einer modularen, skalierbaren Elektrik/Elektronik-Architektur für aktive, passive und integrierte Sicherheitssysteme.

- EAST-EEA – Electronics Architecture and Software Technologies – Embedded Electronic Architecture
  Hier haben sich Fahrzeughersteller, Zulieferer und Tool-Hersteller zusammengeschlossen, um einheitliche Entwicklungsmethoden sowie eine einheitliche Middleware zu spezifizieren.

- JASPAR – Japan Automotive Software Platform and Architecture
  Diese Initiative, die von Toyota und Nissan ins Leben gerufen wurde, ist mit AUTOSAR vergleichbar. Die erste Priorität bei JASPAR haben allerdings Kostensenkungen und weniger technische Aspekte. Zumindest Toyota ist ebenfalls aktives Mitglied bei AUTOSAR, so dass aus JASPAR keine direkte Konkurrenz zu AUTOSAR entstehen sollte.

- MSR – Manufacturer Supplier Relationship
  Diese 1990 gegründete Initiative der Elektrik-/Elektronik-Entwicklungsleiter mehrerer Fahrzeughersteller (Audi, BMW, DaimlerChrysler, Porsche, Volkswagen) und Zulieferer (Bosch, Hella, Siemens VDO) möchte die Zusammenarbeit zwischen OEM und Zulieferern verbessern.

- OSEK/VDX – Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug
  OSEK wurde 1993 von mehreren deutschen Fahrzeugherstellern und Zulieferern als Kooperation gegründet. 1995 fand ein Zusammenschluss mit französischen OEMs statt, die ihren OSEK-verwandten VDX-Standard einbrachten. Der

Standard OSEK/VDX definiert ein API für die zugrunde liegende Systemsoftware, die auf Steuergeräten im Fahrzeug eingesetzt wird.

- ODX – Open Diagnostic Data Exchange
  Der ODX-Standard (ASAM MCD-2D) wurde in einer speziellen Arbeitsgruppe der ASAM entwickelt und steht für einen formalen Beschreibungsstandard, der alle Informationen (Anforderung bzw. Dokumentation), die in der Fahrzeug- und Steuergerätediagnose relevant sind, für die Software-Konfiguration, z.B. zur Bedatung eines Diagnosetesters, zur Verfügung stellt.

2.6 Resümee aus der Geschichte der Kraftfahrzeugelektronik

2.6.1 Architektur und Vernetzung

Aus den vorangegangen Darstellungen wird deutlich, dass Fahrzeughersteller und -zulieferer die Notwendigkeit zur engeren Kooperation erkannt haben. Es ist zu erwarten, dass dadurch in Zukunft zuverlässigere Elektroniksysteme in die Fahrzeuge eingebunden werden [Varchmin05].

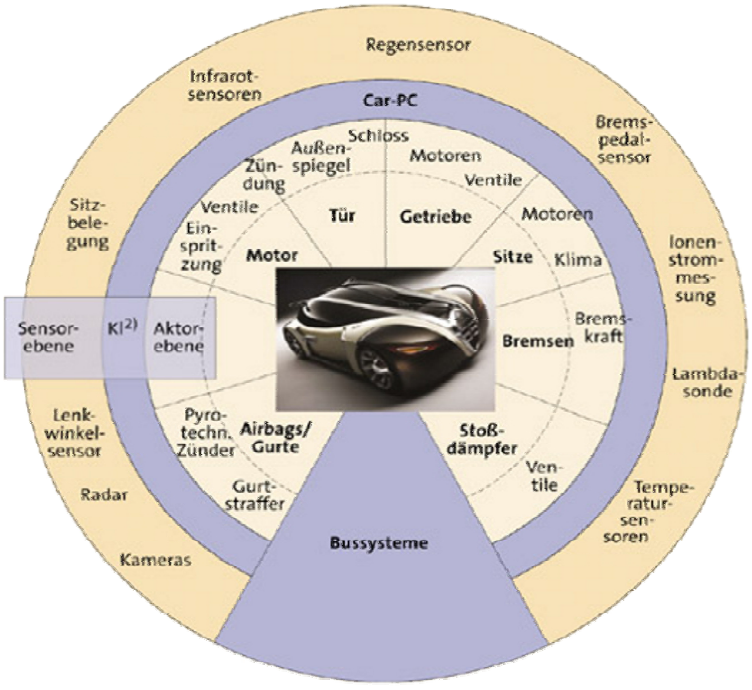

Abbildung 11: Elektronik – In Zukunft vernetzt – Automobil 2010 [Mercer04]

Die Abbildung 9 aus Abschnitt 2.2.5.2 zeigt den starken Anstieg an elektronischen Steuergeräten in den letzten Jahren. Aktuelle Oberklassefahrzeuge können bis zu 70 Steuergeräte und mehr besitzen. Diese Entwicklung wird nicht in dem Maße weiter gehen können. Jedes weitere Steuergerät verursacht Kosten, braucht Zuleitungen, trägt zum Fahrzeuggewicht bei und beansprucht ohnehin schon knappen Bauraum. Da immer mehr Funktionen in Software realisiert werden und die Leistungsfähigkeit der Mikrocontroller stetig zunimmt, ist davon auszugehen, dass neue Funktionen in

vorhandene Steuergeräte integriert werden. Dieser Trend lässt sich am Beispiel des aktuellen 5er BMW belegen, dessen Funktionen in den Bereichen Karosserie- und Sicherheitselektronik sowie Infotainment in weniger Steuergeräte integriert wurden, als es noch im 7er BMW der Fall war [Reichart04].

Ein weiterer zukünftiger Trend ist die zunehmende Vernetzung im Fahrzeug. Dies betrifft die physikalische Vernetzung über Datenbusse genauso wie die logische Vernetzung der Funktionen untereinander. In Abbildung 11 ist dieser Trend auf Grundlage einer Studie [Mercer04] grafisch dargestellt, die als Ergebnis eine vollständige Vernetzung aller Steuergeräte im Fahrzeug bis 2010 sieht und deren Funktionalität allein durch Software bestimmt wird. Die Ringstruktur der Darstellung soll die Vernetzung zwischen Sensorik und Aktorik durch Bussysteme verdeutlichen, während der zentrale Ring die Funktionalität der Software repräsentiert. Die Vernetzung der Funktionen wird zu einem erhöhten Kommunikationsbedarf führen, den geeignete Datenbusse bedienen müssen. Aus Anforderungen nach Echtzeit, Kosten und Sicherheit werden verschiedene Datenbusse zum Einsatz kommen, die zusammen ein heterogenes Kommunikationsnetzwerk bilden und in den nachfolgenden Kapiteln beschrieben sind.

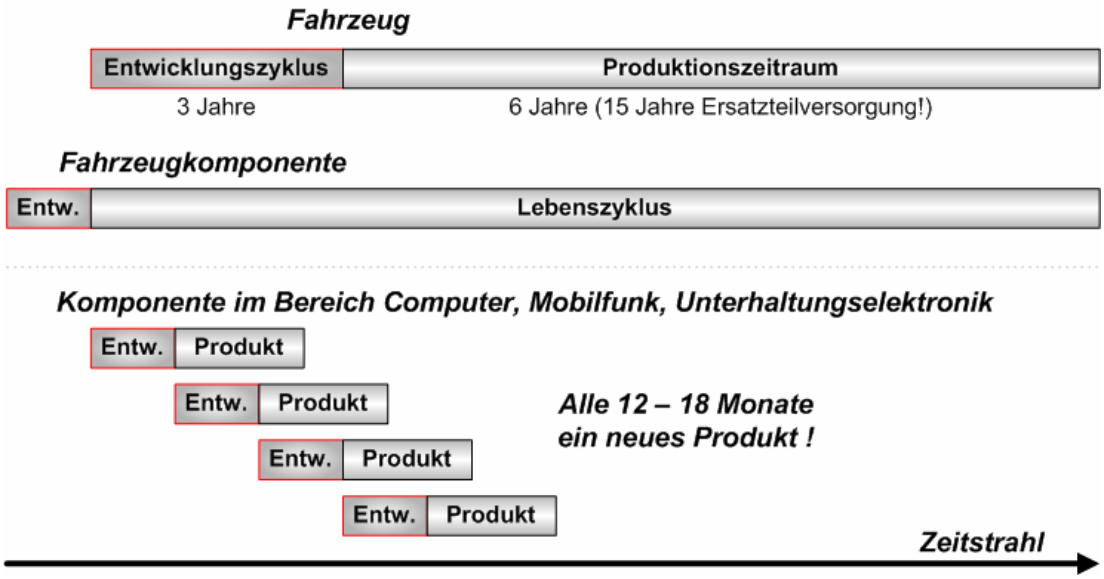

Abbildung 12: Entwicklungszyklen der Automobil- und Consumerelektronikbranche [Teepe00]

Dabei gilt es, den entstehenden Widerspruch zwischen steigender Komplexität des Gesamtsystems und kürzeren Entwicklungszeiten, wie die Abschnitte 2.3 und 2.4 zeigen, zu lösen. Die kurzen und sich teilweise überlappenden Entwicklungs- und Produktlebenszyklen der Computer- und Telekommunikationsbranche, die in Abbildung 12 gegenübergestellt sind, wird man mit dem langfristig orientierten Produkt Automobil nicht erreichen können. Hier gilt es über eine Systemarchitektur und definierte, nach Möglichkeit standardisierte, Schnittstellen die Möglichkeit zu schaffen, dass sich ein Fahrzeug auch über seinen Lebenszyklus von bis zu 15 Jahren in die weiterhin expandierende Welt der Mikroelektroniktechnologien einfügen lässt. Mit Hilfe solcher Schnittstellen lässt sich gleichzeitig das Problem der Vorwärtsbewegung zu größerem Funktionsumfang und der Rückwärtskompatibilität zur problemlosen Ersatzteilversorgung lösen [Teepe00].

2.6.2 Konsequenz für diese Arbeit

Der Begriff der Systemarchitektur wird bei [Hofman01] umfassend beschrieben. Die Aspekte einer Architektur reichen dabei von ihrer Funktion über die Topologie zur Technik bzw. zu den Technologien. Diese Arbeit legt den Schwerpunkt auf die Topologien und Technologien. Die Funktionen spielen dafür im Detail keine Rolle, hier reicht die Forderung, dass prinzipiell jedes Steuergerät mit jedem Steuergerät kommunizieren kann. Es wird davon ausgegangen, dass die standardmäßige Kommunikation, wie sie während des normalen Fahrzeugbetriebs auftritt und vom Automobilhersteller geplant und ausgelegt wurde, funktioniert. Als Bewertungskriterien für eine Architektur sollen die in den vorangegangen Abschnitten vorgestellten Anforderungen dienen. Mittelpunkt dieser neuen Architektur wird der neu entwickelte zeitgesteuerte Datenbus FlexRay sein.

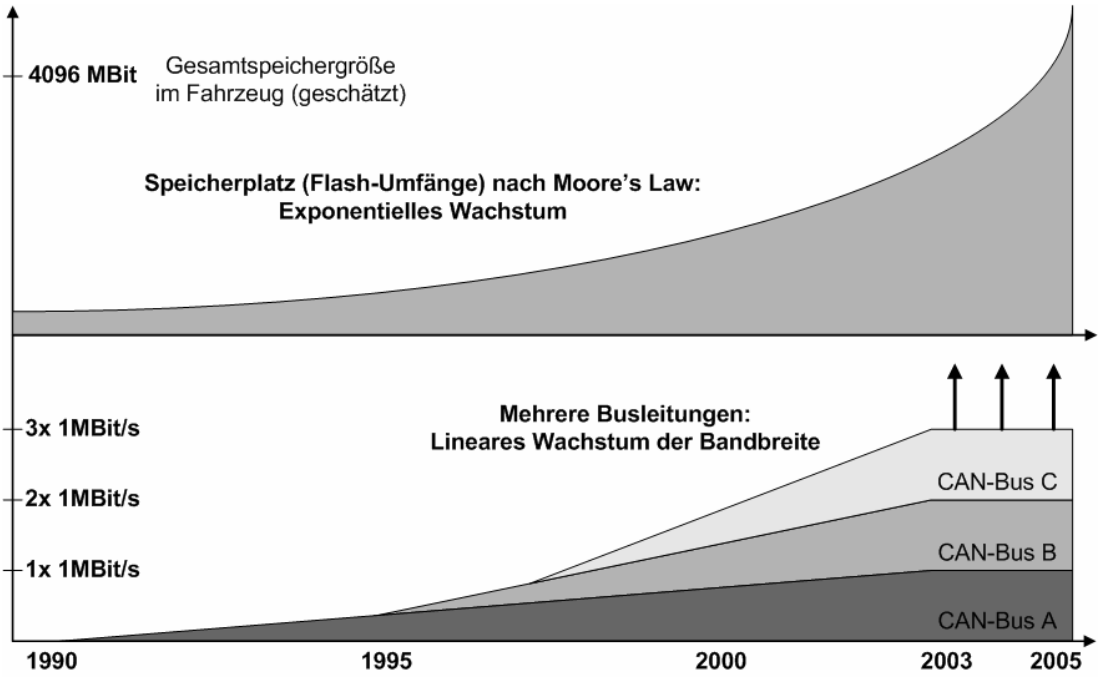

Abbildung 13: Speicher vs. Bandbreite [Heigl04]

Ein wesentlicher Trend der Innovationsphase ist die Vernetzung des Fahrzeugs mit seiner Umwelt, sei es für Car-to-X Communication, Telematikdienste oder der Diagnose. Dafür wird eine definierte Schnittstelle zwischen Fahrzeug-Datennetz und Außenwelt benötigt. Mit einem weiteren Merkmal der Innovationsphase, dem weiter steigenden Softwareanteil im Kraftfahrzeug, wächst auch die Anzahl der unentdeckten Fehler in der Steuergerätesoftware und damit das Bedürfnis, die Software durch Reprogrammieren (Flash-Update) nachträglich verändern zu können. Visionen wie „Software als Produkt“ oder „Softwaretankstelle“, bei denen (Software-)Funktionen auch nachträglich gekauft werden können, sind ebenfalls nur per Flash-Update zu realisieren. Aus diesem Grund wird die hier vorgestellte Architektur einen besonderen Schwerpunkt auf den Diagnosezugang zum Fahrzeug legen und die Applikation „Softwareflashen“ behandeln.

Der in aktuellen Fahrzeugarchitekturen übliche drahtgebundene Diagnosezugang wird den zu erwartenden Datenumfängen nicht mehr gewachsen sein. Abbildung 13 zeigt den Flashumfang, der ähnlich wie Speichermenge und Rechenleistung nach Moore's Law exponentiell wachsen wird. Dem gegenübergestellt ist der CAN-Datenbus, der mit seinem maximalen 1 MBit/s schon für heutige Flashumfänge zu langsam ist. Selbst die parallele Programmierung über mehrere CAN-Datenbusse, sofern dies überhaupt von der Diagnoseschnittstelle unterstützt wird, kann mit dem Wachstum des zu erwartenden Flashumfangs nicht mithalten. Hieraus wird auch deutlich, dass die Diagnoseschnittstelle zukünftig höhere Datenraten als heute üblich unterstützen muss. Zudem ist es nahe liegend, den CAN-Datenbus in speziellen Bereich durch einen Datenbus höherer Bandbreite zu ersetzen. Der hier vorgestellte Ansatz einer Kraftfahrzeugarchitektur verwendet daher FlexRay zur Anbindung der Diagnoseschnittstelle an das Fahrzeugdatennetz. Mit einer zehnmal höheren Datenrate als CAN bietet FlexRay ausreichende Reserven für den Datenzugang in das Fahrzeug. Bei der Realisierung der Diagnoseschnittstelle wird zudem die Möglichkeit des drahtlosen Zugangs zum Kraftfahrzeug untersucht.

Die Entwicklung der Kraftfahrzeugelektronik war bisher von Evolution an Stelle von Revolution geprägt. Daher wird im nächsten Kapitel der aktuelle Stand heutiger Kommunikationsarchitekturen beschrieben, bevor der Ansatz einer zukünftigen Kommunikationsarchitektur auf Basis des FlexRay zusammen mit einem drahtlosen Diagnosezugang vorgestellt wird. Damit lassen sich die Randbedingungen und Vorteile der hier vorgestellten Architektur besser begründen.

3 Datenkommunikation in modernen Kraftfahrzeugen

Dieses Kapitel stellt die Kommunikationsstrukturen und -mechanismen heutiger Kraftfahrzeuge vor. Dazu wird der grundsätzliche Aufbau eines elektronischen Systems im Kraftfahrzeug kurz behandelt. Im Anschluss daran werden die verschiedenen automobilen Datenbusse und die einzelnen Möglichkeiten ihrer Kommunikation untereinander, im folgendem als Kommunikationsarchitektur bezeichnet, vorgestellt.

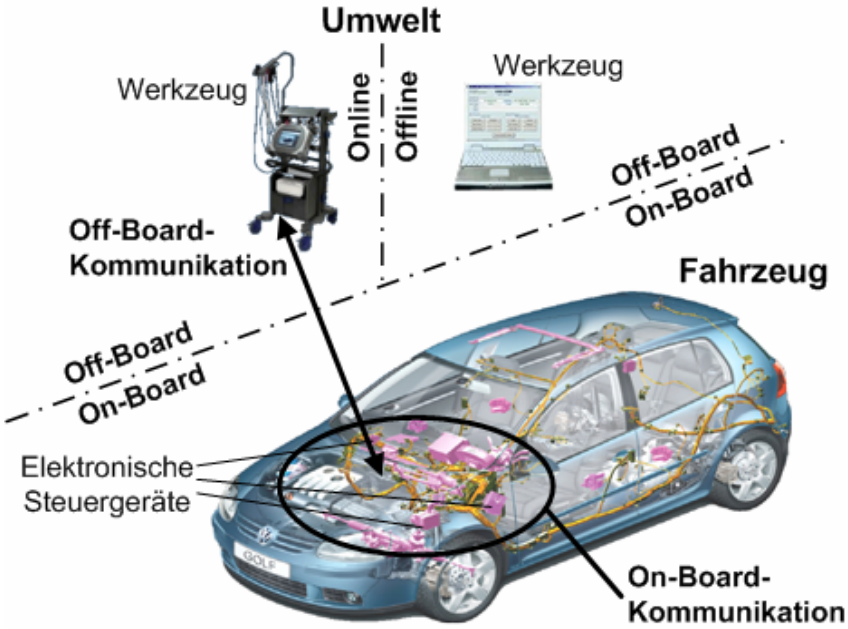

Abbildung 14: Elektronische Systeme des Fahrzeugs und der Umwelt (nach [Schäuffele03])

Die Kommunikation in Kraftfahrzeugen kann in unterschiedliche Typen unterteilt werden (siehe Abbildung 14). Dabei wird die Kommunikation zwischen elektronischen Systemen innerhalb des Fahrzeugs als On-Board-Kommunikation bezeichnet. Die Kommunikation zwischen Systemen des Fahrzeugs und Systemen der Umwelt bezeichnet man dagegen als Off-Board-Kommunikation. Diese kann wiederum zwischen Online- und Offline-Kommunikation unterschieden werden, je nach zeitlicher Relation der Ausführung einer Funktion durch ein System der Umwelt in Bezug zur Ausführung einer Funktion eines Fahrzeugsystems.

Für diese Arbeit ist speziell für die Update-Programmierung die Online-Off-Board-Kommunikation, die über die Diagnoseschnittstelle abläuft, von Bedeutung. Dabei findet die Kommunikation mit einem Steuergerät im Fahrzeug, dem Diagnose- oder Gateway-Steuergerät, statt. Da allerdings nahezu mit jedem elektronischen Steuergerät innerhalb des Fahrzeugs kommuniziert werden muss, läuft der weitere Datenaustausch als On-Board-Kommunikation ab. Bevor auf die einzelnen Details dieser Kommunikation eingegangen wird, folgt ein Überblick über den Aufbau des elektronischen Systems „Kraftahrzeug“.

3.1 Ereignissteuerung vs. Zeitsteuerung

In der Folge wird häufig von zeitgesteuerten Systemen und Echtzeit die Rede sein. Aus diesem Grund werden in diesem Abschnitt kurz die Unterschiede der ereignis- und zeitgesteuerten Systeme erläutert.

3.1.1 Echtzeit

Die Definition bezeichnet Echtzeitsysteme in der Datenverarbeitung allgemein als Rechnersysteme, die mit externen technischen Systemen in einer Wechselwirkung stehen und technische Prozesse steuern oder regeln. Der Echtzeitbetrieb unterscheidet sich dabei von der allgemeinen Datenverarbeitung durch das explizite Hinzutreten der Dimension Zeit. So muss in einem Echtzeitsystem die Verarbeitung der Programme zeitlich mit den im technischen System ablaufenden Vorgängen Schritt halten [Ringler02]. Ein Echtzeitsystem ändert seinen Zustand zu vordefinierten Zeitpunkten. Gleichzeitig muss es auf bestimmte Änderungen in einem definierten Zeitintervall reagieren. Das Ende eines Zeitintervalls wird dabei als Deadline bezeichnet. Kann ein Überschreiten der Deadline temporär akzeptiert werden, wird es in der Literatur als weiche Echtzeit bezeichnet. Führt das Überschreiten der Deadline zu einem Verlust der Systemfunktion, spricht man von einer harten Echtzeit. Verteilte Echtzeitsysteme bestehen aus einer Menge von Rechner-Knoten (Steuergeräte), die über ein Echtzeit-Kommunikationssystem Daten austauschen [Kopetz97].

3.1.2 Ereignisgesteuert

Die meisten der momentan im Fahrzeug eingesetzten Datenbusse sind ereignisgesteuert. Die Nachrichten werden bei Bedarf ausgesandt bzw. empfangen, eine Priorisierung geschieht im Allgemeinen durch die Nachrichtenkennung. Es werden in der Regel nur Nachrichten übertragen, wenn Ereignisse aufgetreten sind (z.B. „Schalter *wurde* gedrückt“). Möchte man nun diese Bussysteme auf zeitkritische und sicherheitsrelevante Anwendungen erweitern, muss ein klarer zeitlicher Bezug (real time – Echtzeit) vom Aussenden der Nachricht zum Empfang und zur Beantwortung gegeben sein. Diese so genannte Latenzzeit muss möglichst gering und definiert sein. Bei einem ereignisgesteuerten System sind die Latenzzeiten nur statistisch bestimmbar. Sollen kurze Antwortzeiten realisiert werden, müssen hohe Datenraten über 500 kBit/s bereitgestellt werden. Gleichzeitig muss für das Erreichen einer geringen Latenz die Busauslastung auf einen Bruchteil der möglichen Buskapazität begrenzt werden. Der Vorteil eines ereignisgesteuerten Systems liegt in der schnellen Reaktion auf asynchrone Vorgänge. Eine genaue Vorhersage (Determinismus), wann welches Datum eintrifft, kann allerdings nicht getroffen werden. Für verteilte Systeme stellt ein ereignisgesteuertes Kommunikationssystem damit ein variables Laufzeitglied dar, was Regelungen über den Datenbus erschwert. Darüber hinaus kommt ein ungenügendes Überlastverhalten bei ereignisgesteuerten Bussystemen hinzu. Ab einer gewissen Menge an Nachrichten werden niederpriore Nachrichten unterdrückt. Gerade für automobile Datenbusse ist dies ein wichtiges Kriterium, da dort meist eine Kausalitätskette vorhanden ist. Beispielsweise würden im Falle eines Unfalls, wo ein schnelles Übertragen von Daten wichtig ist, gerade hier sehr viele

Nachrichten verschickt werden müssen. Diese führen zu einem starken Ansteigen der Busauslastung, was bei ereignisgesteuerten Datenbus von der Verzögerung einzelner Nachrichten bis hin zum Blockieren führen kann.

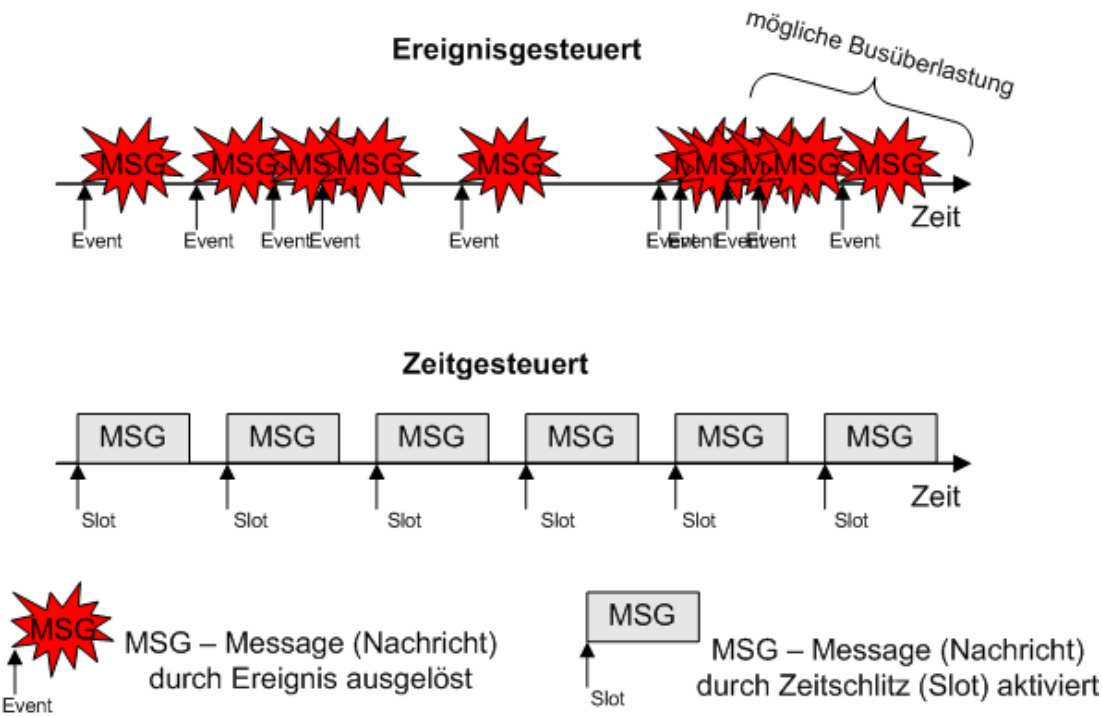

Abbildung 15: Ereignissteuerung vs. Zeitsteuerung

3.1.3 Zeitgesteuert

Im Gegensatz dazu werden bei zeitgesteuerten Systemen Aktionen zu definierten Zeitpunkten ausgeführt. Bezogen auf das Beispiel des Kommunikationssystems bedeutet dies die zyklische Übertragung von Zuständen (z.B. „Schalter gedrückt“) zu definierten Zeitpunkten. Abbildung 15 verdeutlicht diesen Zusammenhang. Durch ihr Verhalten eignen sich zeitgesteuerte Systeme daher besonders für sich wiederholende Vorgänge. Man erhält mit diesen Systemen definierte Latenzzeiten, die beispielsweise für Regelungen als konstante Laufzeitglieder genutzt werden können. Dazu ist ein zeitgesteuertes System per Definition deterministisch, d.h. vorhersagbar und statisch berechenbar. Die wesentliche Eigenschaft eines zeitgesteuerten Systems ist das Vorhandensein einer global synchronisierten Uhrzeit, auf die sich alle Teilnehmer des Systems beziehen. Alle Aktionen sind statisch festgelegt, so dass das System weitgehend konstant über die Zeit ausgelastet ist. Diese statische Festlegung muss schon während der Auslegungsphase des Systems erfolgen, was zeitgesteuerte Systeme relativ unflexibel gegenüber ereignisgesteuerten Systemen macht.

3.2 Elektronisches Steuergerät

3.2.1 Allgemein

Die Basis des elektronischen Gesamtsystems im Kraftfahrzeug bilden einzelne elektronische Steuergeräte, die über eine Vielzahl von Schnittstellen mit den unterschiedlichsten Sensoren und Aktoren verbunden sind. In den Bereich der Sensoren können dabei zusätzlich Sollwertgeber und Benutzerschnittstellen gezählt werden. Diese Sensoren erfassen Betriebsbedingungen wie z.B. Motordrehzahl, Raddrehzahl, Temperatur usw., während Sollwertgeber und Benutzerschnittstellen bestimmte Einstellwerte vorgeben. Sie wandeln diese physikalischen Größen in elektrische Signale um, die durch das elektronische Steuergerät nach bestimmten mathematischen Rechenvorschriften, meist Steuer- und Regelalgorithmen, verarbeitet werden. Darüber hinaus bildet das Steuergerät die Schnittstelle zu anderen Systemen und zur Fahrzeugdiagnose. Es steuert die Aktoren mit elektrischen Ausgangssignalen an, die diese wiederum in mechanische Größen umsetzen.

Die meisten elektronischen Steuergeräte in einem Fahrzeug sind nach diesem Prinzip aufgebaut. Je nach Domänenzuordnung gibt es kleinere Unterschiede und Randbedingungen. Steuergeräte im Antriebsstrang sind sehr leistungsfähig und haben eine sehr große Anzahl an Schnittstellen für Sensoren und Aktoren. Sie werden nah am jeweiligen Aggregat verbaut und ihre räumliche Verteilung im Fahrzeug ist eher gering. Allerdings müssen sie dort besonders rauen Umweltbedingungen wie einem erweiterten Temperaturbereich, Feuchtigkeit und hohen Erschütterungen widerstehen. Dagegen sind Steuergeräte der Karosseriedomäne räumlich weit im Fahrzeug verteilt und übernehmen dabei oft eine Funktion vor Ort. Bei diesen Steuergeräten sind die Übergänge zu intelligenten Sensoren und Aktoren fließend.

Auf die Sensoren und Aktoren wird an dieser Stelle nicht weiter eingegangen. Vielmehr soll eine der wichtigsten Komponenten des Steuergeräts, der Mikrocontroller, betrachtet werden. Er führt die Software aus, die das Verhalten des elektronischen Steuergeräts bestimmt. Der Zusammenhang zwischen Software und Mikrocontroller wird ebenfalls näher beschrieben.

3.2.2 Mikrocontroller

3.2.2.1 Entwicklung der Mikrocontroller

In den 60er Jahren entwickelten Intel und später auch Motorola die ersten Mikroprozessoren, aus denen bis Ende der 70er Jahre die Ein-Chip-Mikrocomputer entstanden. Bekannte Beispiele sind die Mikrocontroller 8048 von Intel und 6800 von Motorola, deren Aufbau auch heute noch Grundlage für viele Mikrocontroller ist. Wie im Kapitel 2 beschrieben, fanden die Mikrocontroller sehr schnell ihren Weg in das Automobil, um dort Steuer- und Regelungsaufgaben zu übernehmen.

Gegen Ende der 80er Jahre wurde zur Abgrenzung gegenüber frei programmierbaren Systemen der Begriff der „Embedded-Systeme“ eingeführt. Embedded-Systeme sind meist für eine bestimmte Anwendung entworfen. Das gesamte Programm, die

Software, inklusive der fahrzeugspezifischen Daten als ROM oder EEPROM (siehe Kapitel 3.2.3) ist fester Bestandteil des Systems und kann nicht durch den Benutzer verändert werden. Die Benutzer haben somit in vielen Fällen nur geringen oder teilweise gar keinen Einfluss auf die Funktionen der Steuergeräte. Im Fahrzeug sind diese Benutzerschnittstellen, sofern überhaupt vorhanden, nur indirekt realisiert und häufig eingeschränkt [Schäuffele03].

Die Mikrocontroller für Embedded-Systeme enthalten meist Derivate von Mikroprozessoren, die ursprünglich für PC-Anwendungen entworfen und später nach entsprechenden Anpassungen und Änderungen als Embedded-Controller vermarktet werden. Mittlerweile existiert allerdings auch eine Reihe von Mikroprozessor-Architekturen, die von Anfang an als Mikrocontroller entwickelt wurden. Beispiele hierfür sind unter anderem der Infineon TriCore, die 8051- oder die C166-Familie. Die Gründe dafür sind im Kapitel 2.3 dargestellt und liegen hauptsächlich darin, dass bei Embedded-Systemen nicht nur die reine Leistung im Vordergrund steht. Vielmehr müssen die Mikrocontroller, speziell im Automobilbereich, einer Vielzahl von mitunter konkurrierenden Anforderungen genügen. Neben der Leistung zählen vor allem Faktoren wie geringe Stromaufnahme, niedriger Energieverbrauch, eine gute EMV, hohe Datendichte, geringe Produktionskosten und eine hohe Anzahl integrierter Schnittstellen für den Datenaustausch mit der Umwelt.

3.2.2.2 Aufbau und Arbeitsweise

Der Mikrocontroller ist ein programmierbarer elektronischer Baustein, der alle Komponenten eines Mikrocomputersystems besitzt. Dazu gehören ein Mikroprozessor (CPU), Ein- und Ausgabeeinheiten, nichtflüchtiger Programm- und Datenspeicher, Datenspeicher als Arbeitsspeicher, ein Bussystem, Taktgenerator sowie eventuell eine Überwachungsschaltung (Watchdog). Sein Aufbau ist in Abbildung 16 schematisch dargestellt.

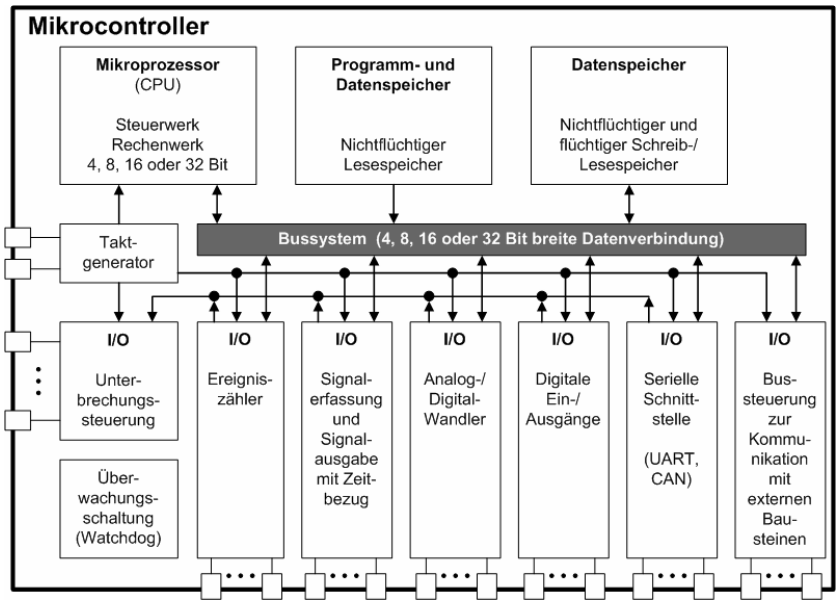

Abbildung 16: Aufbau eines Mikrocontrollers [Schäuffele03]

Der Mikroprozessor ist das Herzstück jeden Rechners bzw. Steuergeräts. Er ist für die Steuerung aller Abläufe und für die Verarbeitung der Daten zuständig. Dazu werden die Befehle des Programms aus dem Speicher gelesen und anschließend dekodiert. Abhängig vom Befehlstyp werden weitere Operanden oder Daten aus dem Speicher geladen und Rechenoperationen durchgeführt. Die Ergebnisse können danach wieder in den Speicher zurück geschrieben werden. Zu den weiteren Aufgaben der CPU neben der Verarbeitung von Daten gehören die Ansteuerung der Peripherie und das Reagieren auf externe Ereignisse (Interrupts). Eine ausführliche Beschreibung zur Funktionsweise von Mikrocontrollern findet sich in [MAT04].

Das für diese Arbeit wesentliche Merkmal ist, dass die Programme und die dazugehörigen Daten für den Mikrocontroller in einem nichtflüchtigen Speicher abgelegt sind. Durch den Einsatz bestimmter Speichertechnologien ist es möglich, diese Programme und Daten nachträglich zu ändern. Aus diesem Grund folgt eine ausführliche Beschreibung der verschiedenen Speichertechnologien, die für den Einsatz im Kraftfahrzeug in Frage kommen.

3.2.3 Speicher von Mikrocontrollern

Prinzipiell wird bei Speichern zwischen den flüchtigen Schreib-Lesespeichern (RAM) und den nichtflüchtigen Festwertspeichern (ROM) unterschieden. Da es bei dieser Betrachtung im Wesentlichen um nichtflüchtigen Festwertspeicher geht, werden die Schreib-Lesespeicher der Vollständigkeit halber nur kurz erwähnt. Die Abbildung 17 zeigt eine Klassifizierung der Speichertechnologien nach [Bosch02]. Eine detaillierte Betrachtung der Flash-Speicher bezüglich des Einsatzes im Kraftfahrzeug wird in Kapitel 4 vorgenommen. In diesem Abschnitt soll eine Übersicht über die verschiedenen Speichertechnologien gegeben werden.

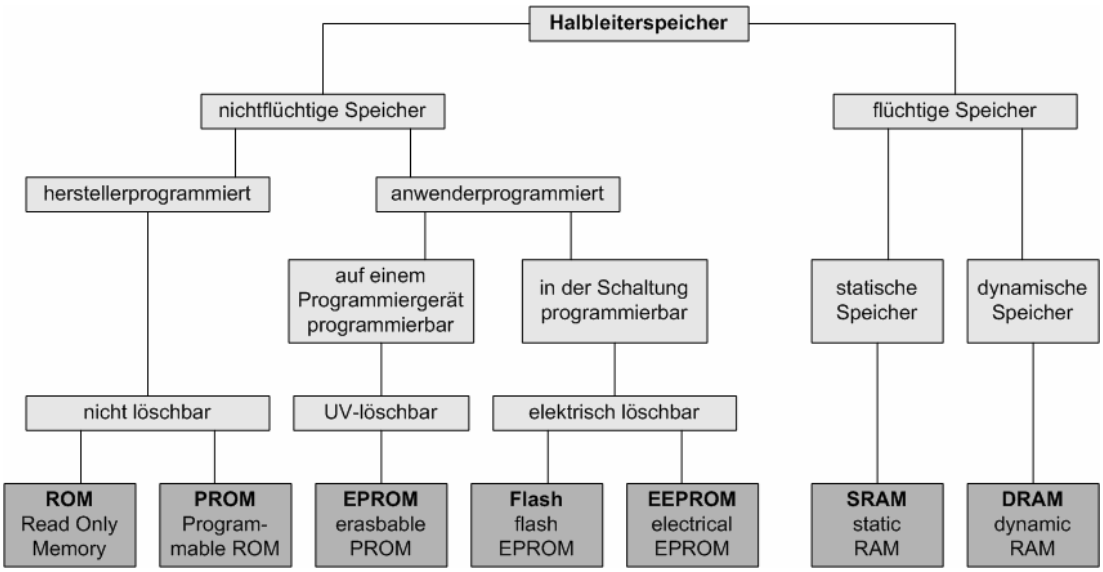

Abbildung 17: Übersicht der Speichertechnologien [Bosch02]

3.2.3.1 Schreib-Lesespeicher (RAM)

Speicher, die im normalen Betrieb beschrieben und gelesen werden können, werden als Schreib-Lesespeicher (RAM) bezeichnet. Diese Kurzzeitspeicher sind flüchtig, d.h. sie verlieren ihren Dateninhalt nach dem Abschalten der Versorgungsspannung. Die Speicherzellen beim statischen RAM bestehen aus bistabilen Schaltungselementen, ähnlich einem Flipflop. Statisches RAM wird einmal beschrieben und behält seinen Speicherinhalt solange die Versorgungsspannung vorhanden ist. Die Speicherzellen von dynamischem RAM bestehen dagegen aus CMOS-Transistoren. Deren Leckströme machen das periodische Auffrischen des Speicherinhalts von dynamischen RAM-Bausteinen notwendig, da sonst deren Speicherinhalt verloren gehen würde. Wird die Spannungsversorgung für das RAM mit einer zusätzlichen Batterie gepuffert, so können Daten auch nichtflüchtig gespeichert werden. In diesem Fall spricht man von nichtflüchtigem RAM (engl. **N**on **V**olatile RAM – NV-RAM).

3.2.3.2 Festwertspeicher (ROM)

Als Festwertspeicher oder ROM gelten alle Speicherschaltungen die ohne Versorgungsspannung ihre Informationen behalten. Diese nichtflüchtigen Speicher sind daher auch als Langzeitspeicher geeignet. Während die Schreib-Lesespeicher als Arbeitsspeicher dienen (d.h. es werden Daten vorübergehend abgelegt), liegen in Festwertspeichern konstante Daten oder Programme vor, wie beispielsweise Tabellen, Kennwertfelder, Parameter oder Programmcode (Steuer- und Regelprogramme). Die verschiedenen ROM-Speicher unterscheiden sich hauptsächlich in der Art der Dateneinbringung (Programmierung) sowie der Möglichkeit der Datenabänderung (Löschung). Man differenziert hierbei prinzipiell zwischen löschbaren und nicht löschbaren Festwertspeichern.

Zu den nicht löschbaren Festwertspeichern zählt das M-ROM (Masken-ROM), das anhand einer vom Hersteller festgelegten und nachträglich nicht mehr modifizierbaren Belichtungsmaske programmiert wird. Der bei der Programmierung festgelegte Datenzustand ist irreversibel. Er kann somit nicht mehr geändert oder gelöscht werden. Da die Erstellung einer Maske für kleine Losgrößen[6] zu teuer ist und ein Fehler beim Entwurf der Maske diese unbrauchbar macht, ist der M-ROM nur für eine Massenproduktion rentabel. Bei großer Stückzahl ist der M-ROM der preiswerteste Speicher, was ihn bislang für die Automobilindustrie interessant machte. Allerdings ist er auch der unflexibelste Speicher, so dass mit steigendem Softwareanteil im Kraftfahrzeug dieser Speichertyp mehr und mehr von löschbaren Festwertspeichern (Flash-Speicher) abgelöst wird. Eine Abwandlung vom M-ROM bezüglich der Programmierung stellt der P-ROM (engl. Programmable ROM) dar. Dieser kann mit einem Programmiergerät elektrisch vom Hersteller oder Anwender genau einmal programmiert werden.

Die EPROM-Speicher (engl. Erasable Programmable ROM) können vom Anwender byteweise elektrisch beschrieben, aber durch Bestrahlung mit ultravioletten Licht (UV-Licht) über ein Fenster nur vollständig, d.h. der gesamte Speicherbaustein, gelöscht werden. Da das Löschen und die Neuprogrammierung mit einem Ausbau des Speicherbausteins verbunden und nur mit Hilfe eines Spezialgeräts möglich ist,

[6] Losgröße bezeichnet die Menge an Gütern, die in einer gemeinsamen Bestellmenge eingekauft wird.

findet er keine weitere Anwendung in Kraftfahrzeugsteuergeräten. Im Gegensatz dazu kann EEPROM-Speicher (engl. Electrically Erasable Programmable ROM) sowohl mit Hilfe eines Programmiergeräts oder auch im verbauten Steuergerät byteweise elektrisch programmiert und gelöscht werden. Verglichen mit dem Flash-Speicher ist, bedingt durch den Speicheraufbau, die Speicherdichte sehr viel geringer und die Programmier- und Löschzeiten relativ lang [Siemers02].

Der Flash-Speicher vereint die Eigenschaften von EPROM und EEPROM, so dass er auch häufig als Flash-EPROM oder Flash-EEPROM bezeichnet wird. Die elektrische Flash-Speicherzelle basiert auf dem Prinzip der EPROM-Zelle, ist jedoch wie das EEPROM auch elektrisch löschbar. Der Flash-Speicher ist in voneinander unabhängige Blöcke unterteilt (z.B. 64 KByte- oder 128 KByte-Blöcke). Dadurch lässt er sich blockweise oder auch komplett[7] löschen, wodurch sich die Löschzeit im Vergleich zum EEPROM erheblich reduziert. Eine ausführliche Beschreibung zu Aufbau und Funktionsweise des Flash-Speichers sowie den speziellen Technologien wie NOR- und NAND-Speichern ist in [Torney04] nachzulesen.

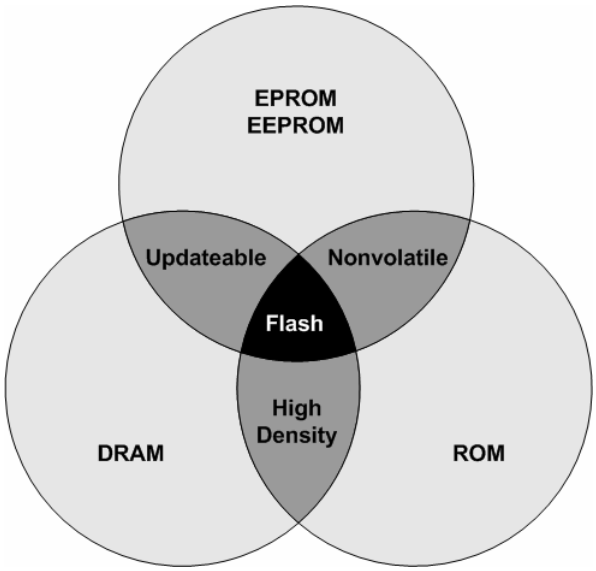

Abbildung 18: Vergleich der verschiedenen Speichertechnologien [Intel02]

Einen Überblick über die Vorteile der verschiedenen Speichertechnologien wie hohe Speicherdichte (engl. High Density), Nichtflüchtigkeit (engl. Non-Volatile) und Wiederbeschreibbarkeit (engl. Updateable) gibt die Abbildung 18. Dort sind die Vorteile des Flash-Speichers, die er neben der kurzen Programmier- und Löschzeit vereint, zusammenfassend dargestellt. Die Flash-Speicher lösen durch ihre Flexibilität zusammen mit den EEPROM-Speichern mehr und mehr die Masken-ROM-Speicher in Kraftfahrzeugsteuergeräten ab. Der EEPROM-Speicher wird aufgrund seines hohen Ausdauerverhaltens und der Möglichkeit der byteweisen Löschung überwiegend zur Datenspeicherung von kleinen und häufig zu überschreibenden Datenmengen (z.B. Kilometerstand) eingesetzt. Flash-Speicher kommen dagegen überall dort zum Einsatz, wo relativ große Datenmengen fest

[7] Das komplette Löschen von Flash-Speicher wird auch als „Chip Erase“ bezeichnet.

gespeichert, aber auch hin und wieder geändert werden müssen, beispielsweise als Programm- und Datenspeicher im Steuergerät.

3.2.3.3 Ort der Speicherintegration

Während die Ablösung der Masken-ROM-Speicher durch Flash-Speicher in vielen Kraftfahrzeugsteuergeräten als sicher gelten kann, bestehen noch Diskussionen bezüglich des Integrationsortes von dem Flash-Speicher auf dem Mikrocontroller. Man unterscheidet bei den Speicherbausteinen zwischen internem Speicher, wie er in Abbildung 16 dargestellt ist, und externem Speicher, der über einen externen Datenbus als eigener Baustein angesprochen wird. Beide Arten haben ihre jeweiligen Vor- und Nachteile, so dass die Fahrzeughersteller je nach gesetzten Schwerpunkten die eine oder die andere Speicherart fordern. Dies macht es besonders für die Speicherhersteller schwierig. Sie müssen mit ihren Speichertechnologien beide Implementierungsarten unterstützen. Dabei ist besonders die Entwicklung des internen Speichers aufwendig, da dieser durch die Integration auf den Mikrocontroller-Chip eng mit diesem verzahnt ist.

Aber gerade der interne Speicher scheint für zukünftige Anforderungen vorteilhafter zu sein, wie es bei [Torney04] im Rahmen dieser Arbeit herausgearbeitet wurde. Externer Speicher benötigt mehr Platinenplatz, Material und zusätzliche Entwurfszeit, was besonders bei dem Trend zu immer höherer Systemintegration auf minimiertem Raum nachteilig ist. Durch die externe Anbindung kommt es darüber hinaus zu erhöhter elektromagnetischer Beeinflussung und unter Umständen zu geringfügig langsameren Zugriffszeiten als beim internen Speicher.

Der wesentliche Nachteil des internen Speichers gegenüber dem externen Speicher liegt in seiner begrenzten Speicherkapazität. Die zurzeit verwendeten internen Speicher haben Kapazitäten bis zu 2 MByte. Der Speicherbedarf von Software liegt im Kraftfahrzeug derzeit zwischen einigen KByte bis zu mehreren MByte, was insbesondere von der Anwendung abhängt. Für viele Anwendungen ist die Speicherkapazität des internen Speichers noch ausreichend. Problematisch sind diesbezüglich Anwendungen aus dem Antriebsstrang (Motor- und Getriebesteuergeräte), während Daten für Multimedia-Anwendungen auf Grund ihres hohen Speicherbedarfs in der Regel von vornherein auf externen Speichern abgelegt sind. Da die Entscheidung für den Integrationsort einige Jahre vor dem Produktionsstart (SOP, engl. Start of Production) fällt, liegt hier die Gefahr, dass die meist knapp kalkulierte Speicherkapazität für größere Softwareänderungen nicht mehr ausreicht. In diesem Fall müsste nachträglich unter erheblichen Kosten- und Zeitaufwand zusätzlicher externer Speicher auf der Platine integriert werden.

Für die Flash-Anwendung ist es prinzipiell unerheblich, welche der beiden Speicherarten zum Einsatz kommen. Ihre Flashroutinen müssen auf die jeweiligen Speicherbausteine abgestimmt sein. Der Anwender dagegen muss die Organisation des Speichers kennen, um die Daten an die richtigen Stellen innerhalb der Flash-Bausteine zu programmieren. Wie dies im Einzelnen geschehen kann und welche speziellen Anforderungen an Flashspeicher gestellt werden, unabhängig ob extern oder intern, beschreibt Kapitel 4.1.

3.2.4 Diagnose

Da sich über ein elektronisches Gerät, wie einem Kraftfahrzeugsteuergerät, von „außen“ und ohne Hilfsmittel nichts über seine korrekte Funktion sagen lässt, entstanden schon sehr früh in der Kraftfahrzeugelektronik-Evolution Überlegungen zur Diagnose der Steuergeräte. Spätestens 1988 wurde durch die CARB (California Air Resources Board) für alle Benzin-Kraftfahrzeuge in Kalifornien die Selbstüberwachung abgasrelevanter Komponenten durch elektronische Steuergeräte (OBD, engl. On Board Diagnose) zur Pflicht. Die Diagnose überwacht ständig das System und seine Komponenten. Auftretende Fehler werden in einem Fehlerspeicher (Flashspeicher) des Steuergeräts abgelegt. Je nach Schwere des detektierten Fehlers kann das Steuergerät weitere Maßnahmen wie Umschaltung in den Notlauf oder Signalisierung im Kombiinstrument ergreifen. Mittlerweile verfügen die meisten elektronischen Steuergeräte im Fahrzeuge über eine eigene elektronische Diagnose und Möglichkeiten zur Diagnosekommunikation.

Erste Umsetzungen der Diagnosekommunikation fanden beim Motorsteuergerät statt, das einen Diagnosestecker herausführte. An diesen konnte ein einfacher Diagnosetester angeschlossen werden und zeigte über Blinkcodes Fehler und Funktion an. Mit zunehmender Vernetzung war diese Lösung nicht lange praktikabel. Viele Steuergeräte waren nicht mehr direkt zugänglich und es konnte nicht für jedes einzelne ein eigener Diagnosezugang vorgesehen werden. Gelöst wurde dieses Problem mit der Vernetzung der Steuergeräte. Über einen angeschlossenen Datenbus, mit dem das Steuergerät mit anderen Systemen kommuniziert, findet auch die Diagnosekommunikation in einem besonderen Betriebsmodus statt. Ebenfalls der Diagnosekommunikation zugehörig ist die Update-Programmierung (Flashen) des Steuergeräts über den Datenbus.

3.2.5 Software

In den Anfängen elektronischer Kraftfahrzeugsteuergeräte wurde die Software für jedes einzelne Steuergerätemodell meist von Grund auf neu programmiert. Als Programmiersprachen kamen anfangs noch Assembler, später auch schon „C“ zum Einsatz. Die Funktionalität der Software war aufgrund der geringen Leistungsfähigkeit der eingesetzten Mikrocontroller begrenzt, was den Programmcode für den Programmierer überschaubar hielt. Die Software beschränkte sich auf ein Steuergerät und es gab nur wenige Wechselwirkungen mit anderen Steuergeräten, was insbesondere das Testen der Steuergerätesoftware einfach machte.

Die Software heutiger Steuergeräte ist hinsichtlich Funktionalität, Umfang und Wechselwirkung mit anderen Steuergeräten um ein Vielfaches komplexer. Mitunter arbeiten verschiedene Zulieferer an einem Steuergerät und seiner Software. Zum überwiegenden Teil wird die Software immer noch in „C“ geschrieben. Aufgrund der hohen Komplexität vollzieht sich hier aber immer mehr ein Wechsel zu Modellierungssprachen wie UML oder direkten Modellierungs- und Simulationswerkzeugen wie Matlab/Simulink. Bei einem großen Teil heutiger Kraftfahrzeugsteuergeräte kommt neben dem eigentlichen Programm ein standardisiertes Echtzeit-Betriebssystem (OSEK) zum Einsatz, welches die Hardware-Ressourcen verwaltet und damit den Programmierer entlastet. Es trennt darüber hinaus die einzelnen

Bereiche wie Kommunikation, Applikation, Netzwerkmanagement usw. und stellt dafür definierte Schnittstellen bereit, was eine Modularisierung der gesamten Steuergerätesoftware ermöglicht. Durch diese Modularisierung wird sehr stark der Anteil der Wiederverwendbarkeit einzelner Softwareteile unterstützt. Solch eine Modularisierung, wie sie in Abbildung 19 dargestellt ist, verringert durch das Wiederverwenden und Austauschen von bereits getesteten Softwareblöcken die Kosten neuer Steuergeräte. Diese Überlegungen gehen so weit, dass die Automobilindustrie viele dieser Softwarefunktionen standardisieren möchte, um eine Austauschbarkeit auch herstellerübergreifend zu ermöglichen (siehe auch Kapitel 2.5.3 und 2.5.4). Gerade Softwarefunktionen zur Datenbuskommunikation oder Update-Programmierung, die notwendig, aber für den Kunden nicht sichtbar und damit nicht wettbewerbsrelevant sind, bieten ein hohes Potenzial für Standardisierungen. Dieser Teil der Software, der nicht direkt zur Applikation und deren Funktionen gehört, wird daher auch Standardsoftware genannt.

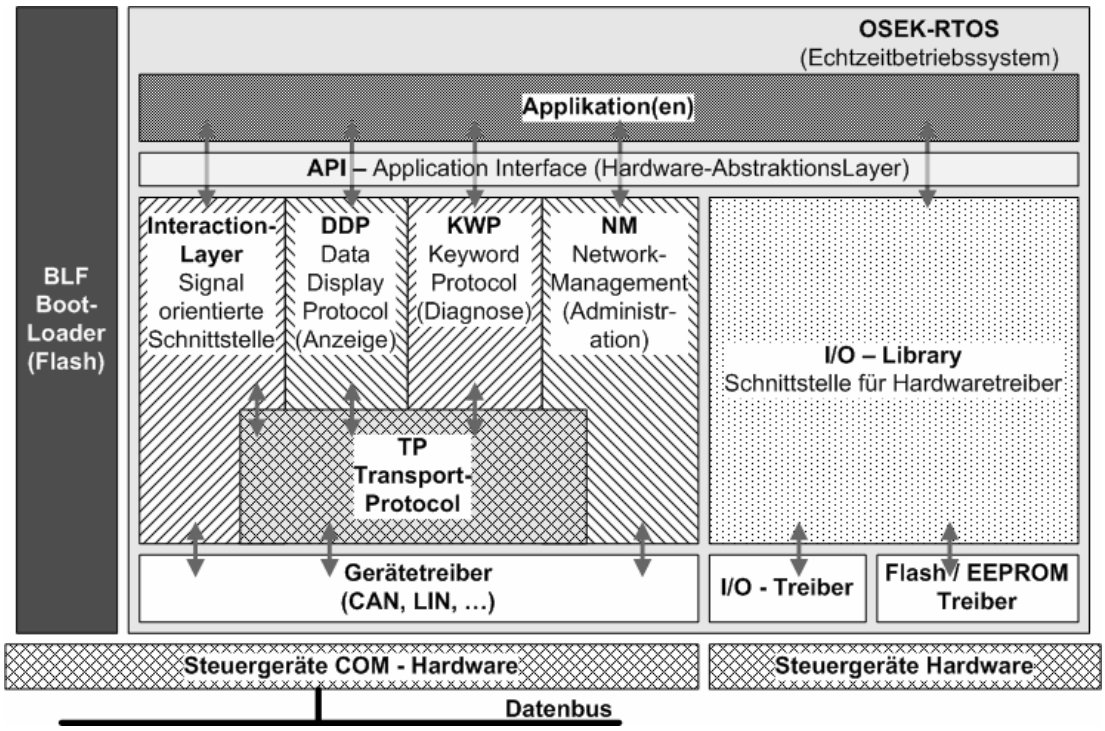

Abbildung 19: (Standard-)Softwarearchitektur mit Bootloader (Flash)

3.2.5.1 Bootloader-Flash (BLF)

Bei Kraftfahrzeugsteuergeräten, die über den Datenbus programmierbar sind, spielt der Bootloader-Flash, auch BLF abgekürzt, eine wichtige Rolle. Bevor eine Neuprogrammierung durchgeführt werden kann, muss vorher der Festwertspeicher, meist Flash, gelöscht werden. Sollte es, bevor diese Neuprogrammierung abgeschlossen ist, zu einem Systemreset oder zu einem Ausfall der Versorgung kommen, befindet sich keine funktionsfähige Applikation mehr auf dem Steuergerät. Die einzige Möglichkeit, solch ein Steuergerät wieder in Betrieb zu nehmen, wäre ein kostenintensiver Aus- und Einbau des Steuergeräts verbunden mit einer manuellen Neuprogrammierung, für die spezielle Werkzeuge notwendig sind. Aus diesem Grund muss für *jeden* Fall sichergestellt sein, dass als Rückfallebene ein

funktionsfähiger Bootloader vorhanden ist. Er ermöglicht auch bei fehlender oder defekter Applikation eine Neuprogrammierung über den Datenbus. Das Vorhandensein eines BLFs erfordert zwingend, dass der Bootloader selber nicht mehr nachträglich programmiert werden darf.

Die wesentliche Aufgabe eines BLF besteht darin, die Applikation auf Korrektheit zu prüfen und zu starten. Sollte der Bootloader feststellen, dass sich keine funktionsfähige Applikation auf dem Steuergerät befindet, wartet er auf eine Anforderung zur Update-Programmierung über den Datenbus. Wenn eine laufende Applikation diese Anforderung erhält und für gültig ansieht (Authentifizierung und richtige Vorbedingungen wie beispielsweise „stehendes Fahrzeug“), wechselt sie zum Bootloader, der dann innerhalb einer bestimmten Zeitspanne auf neue Daten wartet.

Beim weiteren Vorgehen existieren zwei verschiedene Philosophien. Die statische Variante beinhaltet im Bootloader-Flash die Flash-Applikation und damit die eigentlichen Flash-Routinen, mit denen die Update-Programmierung vorgenommen wird. Vorteil hierbei ist, dass die Flash-Routinen aus dem ROM-Speicher ausgeführt werden. Damit kann der vollständige RAM-Speicher zur Zwischenspeicherung der zu programmierenden Daten genutzt werden. Nachteilig wirken sich dagegen die größere Codegröße des Bootloader aus, der in einem eigenen geschützten Flash-Bereich liegen muss sowie die Unflexibilität hinsichtlich späterer Änderungen am Bootloader. Die dynamische Variante hingegen lädt über den Bootloader zuerst die eigentliche Flash-Applikation in das RAM und startet diese. Danach werden die zu programmierenden Daten über den Bus gesendet. Als Nachteile gelten hier die Ladezeit der Flash-Applikation, die die Flashzeit verlängert und der zusätzliche Bedarf an RAM-Speicher für die Flash-Applikation. Gerade bei Steuergeräten mit kleinerem RAM-Ausbau macht sich dies negativ bemerkbar. Die Vorteile liegen dagegen in der Möglichkeit, mit den jeweils aktuellen Flash-Routinen arbeiten zu können und dem Umstand, dass eine Flash-Applikation, die nicht auf dem Steuergerät gespeichert ist, auch nicht auf diesem verändert werden kann (z.B. Datenfehler, Manipulation).

3.2.6 Verteilte und vernetzte Systeme

Wie in den vorangegangenen Kapiteln schon angedeutet wurde, ist in heutigen Kraftfahrzeugen eine Funktion nicht mehr nur auf ein Steuergerät beschränkt. Vielmehr wird beim elektronischen System Kraftfahrzeug von einem verteilten und vernetzten System gesprochen. Nach [Kiencke00] besteht ein verteiltes und vernetztes System aus mehreren Subsystemen, die miteinander kommunizieren, wobei sowohl die Steuerung, als auch die Hardware und die Daten zumindest teilweise dezentral organisiert sind. Die einzelnen Subsysteme werden auch als Knoten bezeichnet und kommunizieren über ein Kommunikationssystem. Die Gesamtheit der Knoten und das Kommunikationssystem bilden zusammen eine Domäne. Ein Kommunikationspfad innerhalb dieser Domäne wird auch als Kanal bezeichnet. Diese Begriffe werden später bei der Vorstellung der Kommunikations-architektur eine Rolle spielen.

Gegenüber zentralen Systemen bieten verteilte und vernetzte Systeme viele Vorteile im Kraftfahrzeug. Sie ermöglichen die räumliche Verteilung von Einzelsystemen, die gemeinsam eine zusammenhängende Funktion darstellen wie beispielsweise die Fensterheber bei der Karosserieelektronik. Vielfach besitzen verteilte und vernetzte Systeme Vorzüge hinsichtlich Erweiterbarkeit und Skalierbarkeit, was zum Beispiel beim Fahrzeug eine große Anzahl von individuellen Sonderausstattungen ermöglicht. Dazu bieten sie große Vorteile bei der ausfalltoleranten Auslegung, was für die Zuverlässigkeit, Verfügbarkeit und Sicherheit von Systemen eine große Rolle spielt.

Gegenüber autonom arbeitenden Einzelsystemen können durch verteilte und vernetzte Systeme häufig auch höhere (Software-)Funktionalitäten realisiert werden. Dies soll am Beispiel des Adaptive-Cruise-Control-System (ACC), einer Weiterentwicklung des klassischen Tempomaten, skizziert werden. Es besteht aus einem Sensor, beispielsweise einem Radarsensor, der Abstände von vorausfahrenden Fahrzeugen oder stehenden Objekten erfasst. Daraus lassen sich Relativgeschwindigkeiten und Winkellagen berechnen, die wiederum zur Klassifizierung von stehenden und sich bewegenden Objekten genutzt werden können. Diese Vorverarbeitung der Sensordaten kann von einem intelligenten Sensor vorgenommen oder muss direkt vom ACC-Steuergerät übernommen werden. An diesem Beispiel lässt sich auch sehen, wie fließend die Grenzen zwischen Steuergeräten und intelligenten Sensoren sind. Aus den so ermittelten Daten berechnet das ACC-Steuergerät die relative Position zu den verschiedenen vorausfahrenden Fahrzeugen und wählt ein „kritisches“ Fahrzeug aus. Durch gezieltes Verzögern und Beschleunigen wird die Längsdynamik des eigenen Fahrzeugs beeinflusst, um einen einstellbaren Sicherheitsabstand konstant einzuhalten. Das ACC-System beeinflusst dazu das Motormoment über das Motorsteuergerät, das Getriebe über das Getriebesteuergerät und das Bremsmoment über das ESP-Steuergerät. ACC ist aus diesem Grund eine antriebs- und fahrwerksübergreifende Funktion und nur ein Beispiel für eines der verteilten Systeme wie sie in modernen Kraftfahrzeugen vorkommen.

Gleichzeitig macht dieses Beispiel auch die Komplexität solcher Systeme deutlich. Stellt sich beispielsweise heraus, dass das Getriebesteuergerät einen Softwarefehler hat und neu programmiert werden muss, so ist sicherzustellen, dass nach dem Flashvorgang ein System wie ACC immer noch einwandfrei funktioniert. Dazu kann es unter Umständen erforderlich sein, verschiedene Steuergeräte per Flashen reprogrammieren zu müssen. Mit geeigneten Kommunikationsarchitekturen, die über definierte Schnittstellen verfügen und einen modularen Aufbau der Software unterstützen, lassen sich derartige Schwierigkeiten vermeiden.

3.3 Automobile Datenbusse

3.3.1 Einführung

Bei der Integration der verschiedenen elektronischen Steuergeräte ins Fahrzeug zeigte sich, dass die meisten dieser Systeme identische Sensorgrößen wie Fahrzeuggeschwindigkeit oder Motordrehzahl benötigen. Umgekehrt können verschiedene Systeme auf dieselben Stellglieder einwirken. Eine elektronische Leerlaufdrehzahlregelung beispielsweise arbeitet so, dass die Drosselklappenstellung entsprechend dem Unterschied zwischen momentaner Leerlaufdrehzahl und Soll-

drehzahl verstellt wird. Ein Eingriff an der Drosselklappe wird aber vom Fahrgeschwindigkeitsregler ebenfalls benötigt. Schon Mitte der achtziger Jahre wurde erkannt, dass elektronische Steuergeräte in steigendem Umfang im Kraftfahrzeug eingebaut werden. Damit lag es damals aus technischen Überlegungen heraus nahe, diese elektronischen Systeme zu einer Zentralelektronik zusammen zufassen [Schwarz78].

Allerdings zeigte sich, dass eine solche Zentralelektronik eine kaum mehr zu beherrschende Komplexität der Hard- und Software zur Folge hatte, die mit jeder neuen Funktion überproportional zunahm. Dazu kam, dass es aus ökonomischer Sicht wenig sinnvoll ist, Kraftfahrzeuge mit wenigen, dafür leistungsfähigen Steuergeräten auszustatten. Damit müssten beispielsweise in Kleinwagen mit geringer Elektronikausstattung die gleichen, in dem Fall überdimensionierten, Steuergeräte eingesetzt werden wie bei voll ausgestatteten Fahrzeugen. Eine preisliche und funktionale Differenzierung zwischen Fahrzeugklassen wäre nur noch schwer möglich. Die Mehrkosten für solche „nicht erwünschten“ Zusatzfunktionen würden nicht vom Kunden bezahlt, sondern müssten von den Automobilherstellern getragen werden.

Daher gingen die Automobilhersteller dazu über, einzelne Funktionen zu modularisieren, um sie nach dem Baukastenprinzip individuell für jedes Fahrzeug nach den Wünschen der Kunden verbauen zu können. Dies führte zu eigenständigen, weitestgehend autark arbeitenden, elektronischen Steuergeräten, die unabhängig von anderen eventuell vorhandenen Systemen im Fahrzeug waren. Somit ließ sich nahezu jede Funktion einem bestimmten Steuergerät zuordnen, was beispielsweise die Diagnose vereinfachte.

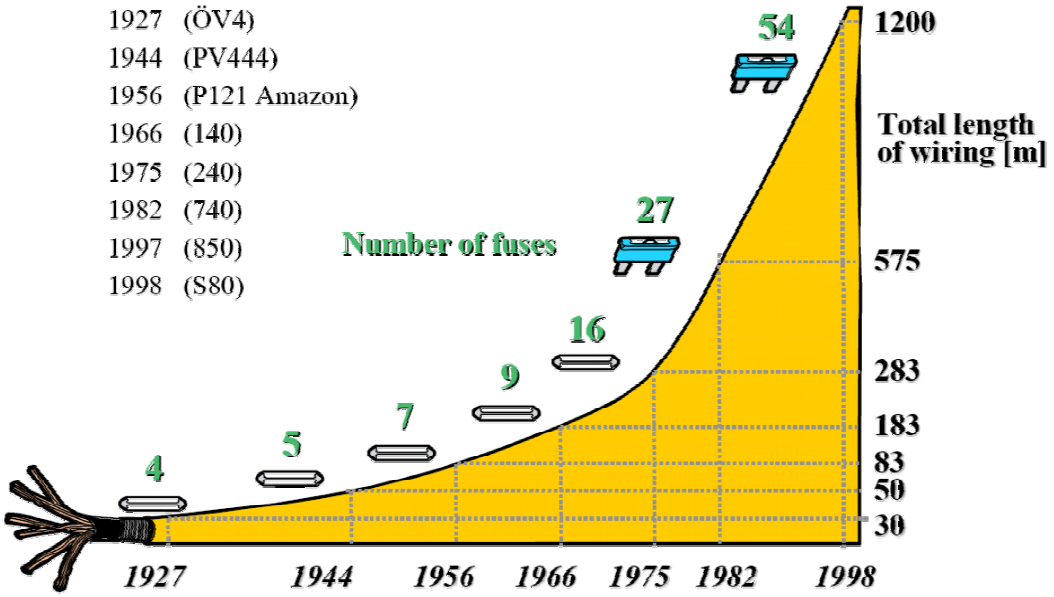

Abbildung 20: Entwicklung der Kabellängen und Anzahl der Sicherungen bei Volvo zwischen 1927 – 1998 [Adolfsson00]

Allerdings mussten die tatsächlich eingesetzten Steuergeräte jeweils über eigene Sensoren inklusive Signalaufbereitung und eigene Aktoren verfügen, was Redundanz und damit Kostenerhöhung bedeutet. Waren die Sensoren und Aktoren außerhalb des Steuergerätes verbaut, mussten sie mit diesem aufwendig verkabelt werden. Jedes einzelne Steuergerät benötigte ebenfalls Zuleitungen zur Energieversorgung. Durch die wachsende Anzahl an Steuergeräten im Fahrzeug wurden aus einzelnen Kabeln Kabelstränge und daraus ganze Kabelbäume. Diese steigerten nicht nur das Gewicht und die Kosten eines Fahrzeuges, sondern führten zu einer sehr hohen Verkabelungskomplexität. Die gesamte Länge des Leitungsstranges wuchs immer stärker, wie Abbildung 20 beispielhaft zeigt. In letzter Konsequenz musste dieser Ansatz versagen, wie auch die damaligen Reklamationszahlen bewiesen, bei denen sich ca. 60% aller Ausfälle auf fehlerhafte Verbindungstechnik, d.h. Steckkontakte, Leitungen, Gehäuse etc. zurückführen ließen [KEhlers86].

Den Widerspruch, das modulare Steuergerätekonzept mit seinen Varianten beizubehalten und trotzdem den Umfang störanfälliger Verbindungstechnik zu reduzieren, lässt sich nur durch ein Kommunikationsnetzwerk lösen. Mit Datenbussystemen ist man in der Lage, die Sensorinformationen universell im Fahrzeug verfügbar zu halten. Gleichzeitig kann eine große Anzahl von Einzelverbindungen durch das Bussystem ersetzt werden, was den Leitungsstrang insgesamt leichter macht, sein Volumen verringert und damit Kosten und Bauraum einspart. Die Kommunikationsnetzwerke bilden die Grundlage für die verteilten und vernetzen Systeme wie sie in heutigen Kraftfahrzeugen zum Einsatz kommen.

3.3.2 Entwicklung der Datenbusse

Gegen Anfang der achtziger Jahre begannen mehrere Automobilhersteller, sich über die Vernetzung von elektronischen Steuergeräten mittels Datenbussen Gedanken zu machen. So sollte ein Datenbus, der die zahlreichen Steuergeräte im Komfortbereich vernetzt, sehr kostengünstig sein. Dabei gab es auch keine hohen Anforderungen an die Datenrate oder an die Ausfallsicherheit. Ein weiterer Datenbus, der die weniger zahlreichen Steuergeräte im Antriebsstrang verbindet, musste dagegen sehr schnell und zuverlässig sein. Für diesen Fall sind höhere Kosten akzeptabel.

Je nachdem, welchem Aspekt die Fahrzeughersteller mehr Bedeutung beimaßen, entstanden sehr unterschiedliche Konzepte für „den“ Fahrzeugdatenbus. Diese Überlegungen resultierten in verschiedenen proprietären Datenbussystemen der einzelnen Hersteller. Beispiele für diese Entwicklungen sind ABUS (Volkswagen), VAN (PSA, Renault), J1850 (GM, Ford), CAN (Bosch, Mercedes) oder K-Bus und I-Bus (BMW) [Angele05].

Anfänglich konnten diese neuen Bussysteme die Erwartungen in sie nur bedingt erfüllen, was der schnellen Integration ins Fahrzeug und der weiten Verbreitung entgegen stand. Vor allem im Kostenbereich hatte die Einführung der Bussysteme zunächst Mehrkosten zur Folge. Wie bei jeder neuen Technologie musste erst die Lernkurve überwunden werden, da sich die Entwicklung eines Fahrzeugs mit einem Bussystem erheblich von dem eines mit herkömmlicher Verdrahtung unterschied. Die hohe Anzahl verschiedener Lösungen verschärfte diese Problematik noch, so dass man nach einem passenden Bussystem suchte. Einzig der CAN-Bus wurde für

die Vielzahl der Anforderungen als geeignet befunden. Allerdings verhinderten vor allen die hohen Kosten des von Bosch und Intel entwickelten Controllerbausteins zunächst seine Einführung im Kraftfahrzeug. Im Bereich der Automatisierungstechnik dagegen stieß der CAN-Bus auf großes Interesse und fand bald mit der für die Automatisierungstechnik erweiterten Spezifikation „CAN Open“ als herstellerneutrale Bus-Lösung größere Verbreitung. Dies führte wiederum dazu, dass Ende der achtziger Jahre einige Halbleiterhersteller begannen, vereinfachte und kostengünstige CAN-Controllerbausteine zu entwickeln, die den CAN-Bus-Einsatz im Kraftfahrzeug zu vertretbaren Kosten ermöglichten. Der erste CAN-Bus in Serie wurde 1992 im Mercedes W140 eingesetzt, wo er das Motorsteuergerät und das Getriebesteuergerät mit dem Kombisteuergerät vernetzte. Kurze Zeit später, Mitte der neunziger Jahre, gaben alle anderen Fahrzeughersteller ihre eigenen Datenbus-Konzepte auf und etablierten bis heute den CAN-Bus als Quasi-Standard der Kraftfahrzeugvernetzung.

Wie wichtig diese Standardisierungen im Bereich Datenbusse für die Automobilindustrie sind, zeigt ein anderes Beispiel. Ein einfacher Diagnosebus hielt noch vor dem CAN-Bus herstellerübergreifend Einzug ins Fahrzeug. 1988 wurde in Kalifornien und später in den gesamten USA per Gesetz vorgeschrieben, dass abgasrelevante Komponenten per OBD (On-Board Diagnose) einheitlich diagnostizierbar sein müssen. Der darauf aufbauende Standard spezifizierte das Kommunikationsprotokoll (die SAE J1850), die Diagnose-Testmodes (SAE J1979), Diagnosecodes (SAE J2012) sowie Stecker- und Pinlayout (SAE J1962), die an den Steuergeräten zugänglich sein mussten. Es war nahe liegend, nicht jedes Steuergerät mit solch einer Diagnoseschnittstelle auszurüsten, sondern diese über einen Diagnose-Datenbus miteinander zu verbinden und einen zentralen Zugang zur Verfügung zu stellen. Allerdings war dieser einfache Datenbus langsam und nie zur Übertragung von Prozessdaten wie beispielsweise der CAN-Bus konzipiert. In Europa wurde diese Diagnoseschnittstelle inklusive Diagnosebus als ISO 9141 standardisiert und befindet sich noch in allen heutigen Kraftfahrzeugen.

Neben diesen beiden Datenbussen existieren weitere Bussysteme für das Kraftfahrzeug, die für spezielle Anforderungen entwickelt wurden. Bevor allerdings die einzelnen der derzeit oder zukünftig im Kraftfahrzeug zum Einsatz kommenden Datenbussysteme hinsichtlich ihrer Eigenschaften bezüglich einer Kommunikationsarchitektur diskutiert werden, ist es sinnvoll, die Anforderungen zu erläutern, die Bussysteme besitzen und speziell im Automobil erfüllen müssen.

3.3.3 Grundlagen von Datenbussystemen

Datenbussysteme können nach verschiedenen Kriterien miteinander verglichen werden. Dabei spielen technische Aspekte wie Leitungslängen, Busteilnehmer und EMV eine Rolle. Aber auch ökonomische Kriterien wie Preis pro Knoten und Gesamtkosten des Bussystems sind wichtig, insbesondere in kostensensitiven Bereichen wie der Automobilindustrie. Letztere Punkte sollen in dieser Betrachtung eine untergeordnete Rolle spielen, da diese sich schwer abschätzen lassen und nur automobile Datenbusse vorgestellt werden, deren Kosten in etwa mit ihrer Leistungsfähigkeit hinsichtlich Übertragungsrate und Fehlerverhalten korrelieren.

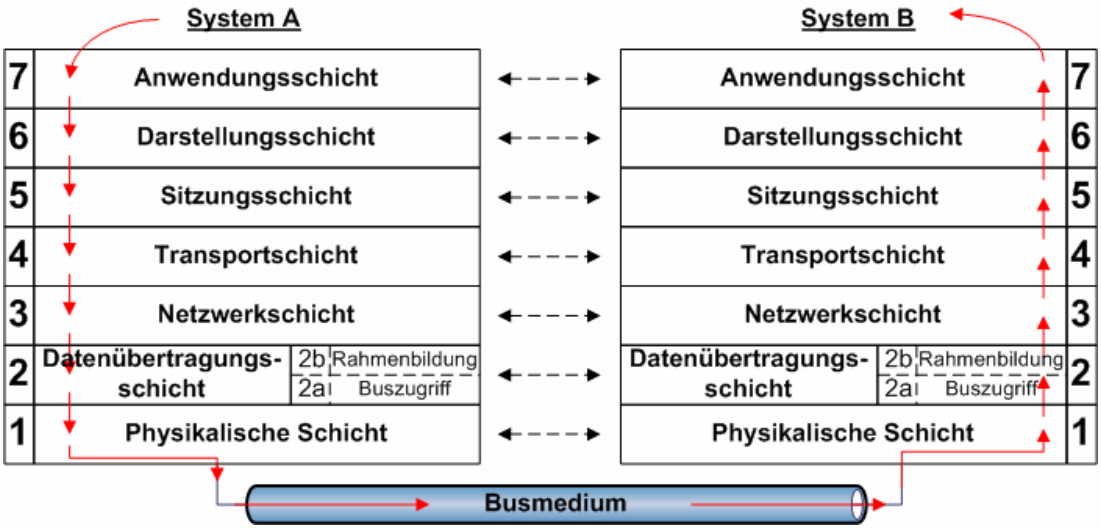

Abbildung 21: ISO/OSI-7-Schichtenmodell

Zur Einordnung und zum Vergleich von Datenbussen bietet sich das ISO/OSI-7-Schichtenmodell in Abbildung 21 als Referenzmodell an. Es beschreibt das Durchlaufen der Daten von sieben logischen Schichten, in denen Funktionen und Protokolle definiert sind, die zur Erfüllung einer bestimmten Aufgabe dienen und die aufeinander aufbauen. Für weiterführende Informationen wird auf [IKF04] verwiesen. Ausgehend von dem ISO/OSI-7-Schichtenmodell werden einzelne Aspekte, die speziell für den Automobilbereich wichtig sind, heraus gegriffen und näher vorgestellt.

3.3.3.1 Übertragungsmedien von Datenbussen

Bei den Übertragungsmedien für Datenbusse unterscheidet man grundsätzlich zwischen drahtgebundenen und drahtlosen Verfahren. Die drahtlose Übertragung gewinnt in den letzten Jahren eine immer größere Bedeutung in der Kommunikationstechnik, was auch für die Automobilindustrie Auswirkungen hat. Der Kunde möchte persönliche Daten seiner drahtlosen Geräte wie PDA[8] oder Telefon mit dem Fahrzeug austauschen und erwartet dafür geeignete drahtlose Schnittstellen vom Fahrzeughersteller. Dieser muss die Sicherheit des Fahrzeugkommunikationssystems gewährleisten und die Architektur der Schnittstellen und des Kommunikationsnetzwerks so flexibel auslegen, dass sie zu den schnellen Innovationszyklen der Kommunikationsbranche kompatibel bleiben (siehe auch Kapitel 2.3). Zusätzlich möchte der Fahrzeughersteller die Vorteile der drahtlosen Übertragung wie Komfort und Verzicht auf schwere und fehleranfällige Mechanik in Bereichen erhöhter Beanspruchung wie Werkstätten zur Diagnosekommunikation nutzen. Dafür sind die Nachteile gegenüber drahtgebundenen Verfahren wie der (Abhör-)Sicherheit, Störanfälligkeit sowie Nichtexklusivität des Busmediums und der damit fest limitierten Bandbreite zu berücksichtigen.

Die gebräuchlichste drahtgebundene Datenübertragung wird über elektrische Leitungen realisiert. Im Kraftfahrzeug versucht man dabei die Anzahl der Leitungsadern aus Gewichts- und Kostengründen möglichst gering zu halten.

[8] PDA = Personal Digital Assistant

Werden beim normalen Fast-Ethernet in der Büroverkabelung 4- und 8-adrige Kabel verwendet, so kommen FlexRay (pro Kanal) und CAN mit zwei Adern aus. Diese sind verdrillt und machen die Datenübertragung weniger störanfällig. Bei „langsameren" Sensordatenbussen wie LIN nutzt man die Fahrzeugmasse als Rückleiter und benötigt nur noch einen einzigen Leiter. Bei zweiadrigen automobilen Datenbussen verwendet man wegen der besseren Resistenz gegenüber externen Störungen Differenzsignale [IKF04]. Die Vorteile der elektrischen Verkabelung sind die langjährigen Erfahrungen mit dieser Verbindungstechnik, ihre Abhörsicherheit, ihre Störresistenz und die hohen Übertragungsraten, die sich damit erzielen lassen. Nachteilig wirken sich dagegen Gewicht, Verkabelungskomplexität und die elektromagnetische Verträglichkeit (EMV – umfasst Störimpulseinstrahlung und -abstrahlung) aus, die insbesondere im Kraftfahrzeug hohen Anforderungen genügen muss.

Gerade die EMV lässt sich mit zunehmenden Datenraten immer schwerer beherrschen. Die Störabstrahlung wird bei höheren Taktraten auf den elektrischen Kabeln immer stärker, und gleichzeitig wirkt sich eventuelle Störeinstrahlung auf die schmalbandigen Signalimpulse immer negativer aus [Habiger98]. Daher setzt man in den letzten Jahren auf eine optische Datenübertragung mittels Glasfaserkabel, wie es der MOST-Datenbus nutzt. Den Vorteilen wie geringe elektromagnetische Störanfälligkeit und hoher Datenrate stehen hier die Nachteile der Kosten und des Aufwandes bei Verkabelung und Reparaturen gegenüber.

3.3.3.2 Topologien von Datenbussen

Die beste Verbindung zwischen zwei Kommunikationspartnern bildet die Punkt-zu-Punkt-Verbindung (Abbildung 22 a). Sie besitzt eine definierte elektrische Umgebung und eine ideale Terminierung, so dass keine Reflektionen auf der Leitung auftreten können. Allerdings sind an einen Datenbus in der Regel mehr als nur zwei Kommunikationspartner angeschlossen. Die Topologie eines Datenbusses bezeichnet die Art und Weise, wie seine Kommunikationsteilnehmer miteinander elektrisch verbunden sind. Jeder Datenbus ermöglicht eine oder mehrere bestimmte Topologien.

Der CAN-Bus beispielsweise erweitert die Punkt-zu-Punkt-Verbindung zu einer Linienstruktur (siehe Abbildung 22 b). Die zusätzlichen Busteilnehmer verschlechtern die elektrischen Eigenschaften des Bussystems, was zu Beschränkungen hinsichtlich Länge und Anzahl der Stichleitungen sowie der Gesamtlänge führt. Ein Defekt des Busmediums verhindert darüber hinaus die gesamte Buskommunikation[9]. Außerdem besteht die Gefahr, dass bei Ausfall eines Teilnehmers dieser unter Umständen die gesamte Buskommunikation stört. Der MOST-Bus nutzt im Gegensatz zum CAN-Bus eine Ringstruktur (Abbildung 22 c). Für diese gelten allerdings die gleichen Nachteile, dass Defekte am Busmedium oder den Teilnehmern das gesamte Bussystem stören.

[9] Eine Ausnahme ist der Fault Tolerant CAN-Bus, der bei bestimmten Defekten (Bruch einer Busader) die Kommunikation weiter aufrechterhalten kann.

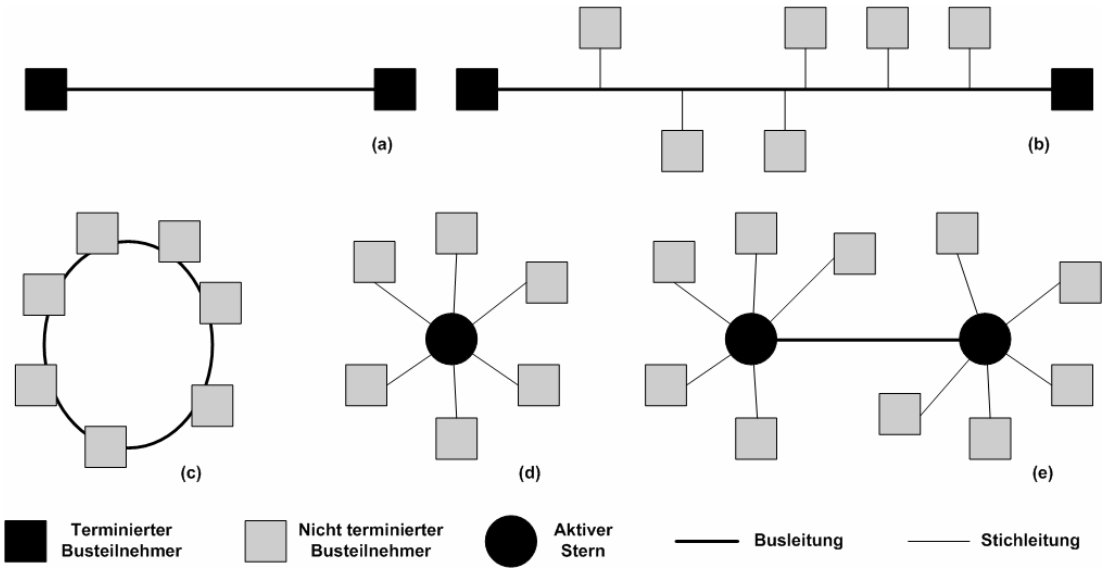

Abbildung 22: Datenbus-Topologien

Diese Nachteile lassen sich mit einem aktiven Stern vermeiden (Abbildung 22 d). Defekte an Busmedien und Teilnehmern sind für das Gesamtnetz nicht mehr kritisch. Darüber hinaus bietet ein aktiver Stern alle Vorteile der Punkt-zu-Punkt-Verbindung, was ihn insbesondere für Echtzeitsysteme geeignet erscheinen lässt. Für sicherheitskritische Systemen stellt solch ein aktiver Stern allerdings einen „Single Point of Failure“ dar. Der Ausfall des aktiven Sterns führt zum Ausfall des Gesamtsystems. Ein weiterer Nachteil ist der erhöhte Verkabelungsaufwand. Dieser lässt sich bei räumlich verteilten Systemen durch kaskadierte aktive Sterne (Abbildung 22 e) in akzeptablen Grenzen halten. FlexRay erlaubt unter bestimmten Randbedingungen bis auf die Ring-Topologie jeder der hier vorgestellten Topologien. Dazu sind auch Mischformen der einzelnen Topologien möglich. FlexRay bietet somit den Automobilherstellern bezüglich der Netztopologie größtmögliche Flexibilität.

3.3.3.3 Zugriffsverfahren bei Datenbussen

Durch einen Datenbus besteht eine direkte Verbindung zwischen allen Teilnehmern. Es existiert ein einziger Übertragungsweg, der eine elektrische Leitung, ein Lichtwellenleiter oder ein Funkkanal sein kann. Durch die Verbindungsstruktur wird gewährleistet, dass alle Teilnehmer auf dem Übertragungskanal mithören können. Probleme ergeben sich jedoch, wenn mehrere Teilnehmer gleichzeitig senden wollen. Diese lassen sich beispielsweise über Multiplexverfahren lösen, in dem beim Frequenzmultiplex, dem FDMA (Frequency Division Multiple Access) jedem Teilnehmer ein eigener Frequenzbereich und damit ein eigener Kanal zugeordnet wird. Beim Zeitmultiplex, dem TDMA-Verfahren (Time Division Multiple Access) dagegen wird das Übertragungsmedium den einzelnen Teilnehmern zeitlich gestaffelt hintereinander zugeteilt. Dieses Verfahren ist insbesondere für kabelgebundene automobile Echtzeit-Datenbusse wie FlexRay oder TTP gebräuchlich.

Die Zeitmultiplexverfahren unterscheiden sich durch die Art des Zeitrasters. Beim Zeitmultiplex mit festem Zeitraster wird jedem Teilnehmer der Bus zu einem

bestimmten Zeitpunkt zugeordnet, unabhängig davon, ob er senden will oder nicht. Die Aufteilung der Zeitfenster und deren Zuordnung zu den Teilnehmern sind statisch festgelegt. Die Auslastung des Datenbusses ist bei diesem Verfahren konstant und vorhersagbar, muss aber für den Worst Case[10] ausgelegt sein. Ökonomischer sind die Zeitmultiplexverfahren mit einem bedarfsabhängigen Zeitraster. Dabei wird das Übertragungsmedium nur zugeteilt, wenn auch wirklich eine Übertragungsanforderung vorliegt. Der Datentransfer wird somit bei normaler Busauslastung schneller abgewickelt. Allerdings ist hierfür wieder eine Zuteilung des Busmediums erforderlich, die von einer übergeordneten Instanz, einem Master, vorgenommen werden muss. Bei Bussystemen ohne eigenen Master erfolgt der Zugriff auf den Datenbus über eine Busarbitrierung[11]. Dabei regeln statisch vergebene Prioritäten, die entweder Teilnehmern oder aber Nachrichten zugeordnet sein können, den Buszugriff. Darüber hinaus existieren weitere, spezielle Zugriffsverfahren wie das Token-Passing-Verfahren oder das Prinzip des Delegated Token, die im Einzelnen bei [Färber95] nachzulesen sind.

Im Gegensatz zu den zeitgesteuerten Datenbussen verwenden ereignisgesteuerte Datenbusse wie CAN ein zufälliges Buszugriffsverfahren wie CSMA (Carrier Sense Multiple Access). Dabei hört jeder Teilnehmer vor dem Senden das Busmedium ab, ob dieses frei oder belegt ist. Durch die endliche Signalausbreitungsgeschwindigkeit auf dem Datenbus kann es dennoch zu Kollisionen kommen. Beim klassischen CSMA wird dies erst nach dem Senden der kompletten Nachricht erkannt, z.B. durch ein nicht gesetztes Paritätsbit oder einem CRC-Fehler. Durch das CSMA/CD-Verfahren wie es beim klassischen Ethernet verwendet wird, wobei „CD“ für Collision Detection steht, hört der Sender gleichzeitig auf dem Bus mit, ob seine Daten richtig übertragen werden. Damit kann eine Kollision bereits früher erkannt und der Bus besser ausgelastet werden. Noch bessere Resultate können mittels des CSMA/CA-Verfahrens erzielt werden, wie es der CAN-Bus einsetzt. Das „CA“ steht für Collision Avoidance und stellt sicher, dass erst gar keine Kollisionen auftreten können. Die bitweise Arbitrierung der Nachrichten-ID [Etschberger94] erlaubt damit die Priorisierung der Nachrichten(inhalte) anstelle der einzelnen Busteilnehmer.

Beide grundsätzlichen Zugriffsverfahren, zeit- und ereignisgesteuert, haben ihre Vor- und Nachteile. Der Aufwand und die Kosten sind bei zeitgesteuerten Datenbussen vergleichsweise hoch. Sie verlangen einen ganzheitlichen Ansatz beim Entwurf des Kommunikationssystems, was allerdings wiederum den modernen Entwicklungsansätzen der Automobilindustrie entgegen kommt. Im Gegenzug erhält man dafür ein sehr gutes Überlastverhalten und ein deterministisches[12] Zeitverhalten. Man spricht von deterministischem Zeitverhalten, wenn der Zeitraum, in der eine Nachricht übertragen wird, garantiert werden kann. Besonders für Echtzeitfunktionen von verteilten und vernetzten Systemen ist dies eine wichtige Voraussetzung, um Regelungen über das Kommunikationssystem durchführen zu können. Damit werden zukünftige Anwendungen wie beispielsweise die X-by-Wire-Technologie erst möglich. Aber auch die ereignisgesteuerten Bussysteme behalten ihre Daseinsberechtigung im Kraftfahrzeug. Hier wird sich der Trend der heterogenen

[10] Worst Case – engl. schlechtester oder ungünstigster (anzunehmender) Fall, der eintreten kann
[11] arbiter = Schiedsrichter
[12] deterministisch = abgeschlossen, begrenzt, *hier auch* vorhersagbar

Kommunikationsnetze aus zeit- und ereignisgesteuerten Datenbussen im Kraftfahrzeug weiter fortsetzen.

3.3.3.4 Übertragungsrate

Eines der wichtigsten Leistungsmerkmale für Datenbusse ist ihre Übertragungsrate, d.h. die maximal übertragbare Datenmenge pro Zeiteinheit. Diese wird üblicherweise in Byte pro Sekunde oder Bit pro Sekunde angeben. Die Übertragungsrate kann in eine Brutto- und in eine Netto-Übertragungsrate unterschieden werden. Für die Applikation ist die Netto-Übertragungsrate entscheidend. Sie ergibt sich, indem von der üblich angegebenen Brutto-Übertragungsrate die Protokolldaten wie Header und Trailer abgezogen werden. Das Verhältnis von Netto- zu Brutto-Übertragungsrate wird im Wesentlichen von der Art der Nachrichten und der maximal möglichen Nachrichtenlänge des Bussystems bestimmt. Gerade bei kleinen Nachrichtenlängen steigt der Overhead überproportional an, wie in Tabelle 1 zu sehen ist.

Durch den CAN-Bus hat sich dabei ein Nachrichtenformat in der Fahrzeugvernetzung etabliert, das eine Nachrichtenlänge zwischen null und acht Bytes erlaubt. Sie ist für Prozessdaten (z.B. Geschwindigkeit oder Drehzahl), wie sie im Kraftfahrzeug üblich sind, ausgelegt und geeignet. Eine Ausnahme bildet der MOST-Bus, der speziell für die Übertragung von Multimediadaten konzipiert ist. Mit Nachrichten bis zu acht Bytes können die meisten der im Fahrzeug anfallenden Daten in einem guten Verhältnis zwischen Brutto- und Netto-Übertragungsrate übertragen werden. Für zu übertragende Daten, die länger als die maximal möglichen acht Bytes sind, kommen Transportprotokolle zum Einsatz (siehe Kapitel 4.4.2). Diese unterteilen die Daten senderseitig in einzelne Segmente, übertragen diese und kümmern sich empfangsseitig um die richtige Zusammensetzung der Datensegmente. Der Vorgang geschieht, wie im ISO/OSI-7-Schichtenmodell vorgesehen, für die Applikation transparent. Allerdings vergrößert sich durch das zusätzliche Transportprotokoll der Overhead zusätzlich.

Datenbus	Nachrichtenlänge (gesamt)	Nutzdatenlänge (maximal)	Overhead
LIN	102 Bit[13]	64 Bit	**~ 37 %**
CAN (standard)	111 Bit	64 Bit	**~ 42 %**
CAN (extended)	129 Bit	64 Bit	**~ 50 %**
MOST	512 Bit	480 Bit[14]	**~ 6 %**
FlexRay	262 Byte	254 Byte	**~ 3 %**

Tabelle 1: Protokoll-Overhead automobiler Datenbusse (idealisiert)

Besonders ineffizient für die Übertragungsrate macht sich die geringe Nachrichtenlänge beim Flashen bemerkbar. Denn dabei müssen typischerweise Daten in einer Größenordnung im Kilobyte- bis in den Megabyte-Bereich übertragen werden. Die

[13] Der Interframe-Space ist bei LIN nicht spezifiziert und wird hier mit einer Bitzeit angenommen.
[14] Ohne Control-Frame (16 Bit), da dort nur spezifische Daten übertragen werden.

Folge ist eine wesentlich verlängerte Flashdauer. Als Ausweg bieten sich zukünftige Datenbusse an, die eine höhere Datenrate und flexiblere Nachrichtenlängen unterstützen wie beispielsweise FlexRay. Am Ende müssen allerdings die Flashdaten bei CAN-Steuergeräten über den CAN-Bus übertragen werden. Hier können geeignete Flash-Strategien, wie sie in den späteren Abschnitten noch diskutiert werden, helfen, die Flashdauer zu verkürzen oder zumindest effizienter zu gestalten (z.B. Zwischenspeicherung von Flashdaten im Fahrzeug).

3.3.3.5 Klassifizierung von automobilen Datenbussen

Die SAE (engl. Society of Automotive Engineers) hat mit dem Standard J1213/1 die Datenbusse, die im Kraftfahrzeug eingesetzt werden, in drei verschiedene Klassen eingeteilt. Die Einteilung geschieht im Wesentlichen durch die maximalen Übertragungsraten der einzelnen Datenbusse. Eng verknüpft mit den Übertragungsraten sind die Kosten eines solchen Datenbusses, die üblicherweise in Kosten pro Teilnehmer (Knoten) angegeben werden. Einige der automobilen Datenbusse wie beispielsweise CAN können je nach Auslegung mehrerer dieser SAE-Klassen angehören. Durch den Zusatz der Klasse wird die Einordnung kenntlich gemacht, z.B. CAN/B oder CAN/C.

- Class A: Geringe Übertragungsrate (bis 10 kBit/s) – Einsatz im Komfortbereich und als Sensorbusse für kleine Datenmengen
- Class B: Mittlere Übertragungsrate (10 kBit/s bis 125 kBit/s) – Einsatz im Komfortbereich für allgemeine Datenübertragungen
- Class C: Hohe Übertragungsrate (125 kBit/s bis 1 MBit/s und größer) – Einsatz in Systemen mit Echtzeitanforderungen wie dem Antriebsstrang

Gemäß dieser Spezifikation würden der MOST und der FlexRay in die Class C-Netzwerke eingeteilt. Zumindest für den MOST hat sich auf Grund seiner hohen Datenübertragungsrate von 24,8 MBit/s die Bezeichnung als Class D-Netzwerk etabliert. Diese Class D-Netzwerke zeichnen sich durch eine sehr hohe Datenübertragungsrate aus und werden zur Übertragung von Multimediadaten genutzt. Sie sind allerdings kein Bestandteil der SAE-Spezifikation. In den folgenden Abschnitten werden die wichtigsten Datenbusse für das Kraftfahrzeug einzeln vorgestellt und am Ende miteinander verglichen.

3.3.4 K-Line

Der Datenbus ISO 9141, auch Diagnose-Bus, ISO-K oder K-Line genannt, ist ein serieller Ein- oder Zweidrahtbus, bei dessen Auslegung Robustheit und Flexibilität wichtiger waren als Geschwindigkeit, die maximal 10,4 kBit/s[15] betragen kann. Er war nie für die Übertragung von Prozessdaten konzipiert und wurde dafür auch nicht eingesetzt. Technisch beruht dieser Datenbus auf der SAE-Norm J1850[16] und wurde

[15] In der ISO 9141 ist die Geschwindigkeit des Diagnosebus bis max. 10,4 kBit/s spezifiziert, in der Praxis sind jedoch Übertragungsraten bis 19,2kBit/s problemlos möglich. BMW nutzt diese Schnittstelle seit dem E65 sogar mit 115 kBit/s.

[16] Die SAE J1850 definiert zwei Varianten. Eine mit 41,6kBit/s, die auf dem SCP-Protokoll von Ford basiert und eine mit 10,4 kBit/s mit variabler Pulsbreite, die mit der ISO 9141 identisch ist.

ursprünglich als Diagnosebus (OBD) eingesetzt. Die K-Line spielt in heutigen Kraftfahrzeugen so gut wie keine Rolle mehr und degeneriert immer weiter zur reinen OBD-Schnittstelle als kleinsten gemeinsamen Standard. Die Diagnosekommunikation wird in heutigen Kraftfahrzeugen bereits über CAN geführt. Einzig der kostengünstige Physical Layer der ISO 9141 findet in anderen Datenbussystemen wie beispielsweise LIN oder TTP/A weitere Anwendung.

3.3.5 Controller Area Network (CAN)

Der am häufigsten eingesetzte Datenbus im Kraftfahrzeug ist der CAN-Bus [CAN]. Er gilt als Quasi-Standard bei der Vernetzung im Automobil. Das „CAN-Protokoll" meint in aller Regel die ersten beiden Schichten des ISO/OSI-7-Schichtenmodells, die für CAN standardisiert sind. Auf Details wie Nachrichtenformat oder Arbitrierung wird an dieser Stelle nicht weiter eingegangen. Diese können in der Literatur nachgelesen werden [Etschberger94], [IKF04], [Detering04]. Der CAN-Bus existiert in zwei unterschiedlichen Ausführungen, die sich in ihren möglichen Geschwindigkeiten und ihrem Physical Layer unterscheiden. Diese beiden Varianten, Low-Speed und High-Speed, nutzen das gleiche Nachrichtenformat. Für die weitere Betrachtung der Kommunikation ist, abgesehen von der Übertragungsgeschwindigkeit, die Art des CAN-Busses unerheblich. Einzig für die Architekturen stellen die beiden Varianten des CAN-Busses zwei unterschiedliche Bussysteme dar.

Darüber hinaus existiert eine CAN-Protokoll-Erweiterung. Neben dem Protokoll CAN 2.0A-Standard, auch Standard-CAN genannt, gibt es noch einen CAN 2.0B-Standard. Dieser wird als Extended-CAN bezeichnet und ersetzt das 11 Bit lange Identifierfeld durch 29 Bit. Diese Extended-Version wurde für amerikanische Trucks benötigt, bei denen die 2048 Nachrichten, die mit 11 Bit adressiert werden können, nicht mehr ausreichten. Innerhalb dieser Arbeit ist mit CAN immer der Standard-CAN gemeint.

3.3.5.1 Low-Speed (Fault Tolerant – FT-CAN)

Der Low-Speed-CAN-Bus (FT-CAN) ist als ISO 11898-1 (Data Link Layer) standardisiert und wird in zwei Klassen unterteilt, die sich im Physical Layer unterscheiden. Als Eindrahtbus (SingleWire-CAN – SW-CAN) kommt er als GM-Lan bei General Motors zum Einsatz. Der Physical Layer für den SW-CAN ist in der SAE 2411 spezifiziert und erlaubt eine maximale Übertragungsgeschwindigkeit von 41,6 kBit/s. Stärker verbreitet ist allerdings die Zweidrahtausführung, die in der ISO 11898-3 standardisiert ist und mit unabgeschlossener Busleitung arbeiten kann. Damit können Bustreiber geringer Leistung eingesetzt und der Ruhestromverbrauch niedrig gehalten werden. Darüber hinaus beschränkt sich die Bustopologie nicht auf eine reine Linientopologie mit geringer Abzweiglänge. Die maximale Übertragungsgeschwindigkeit liegt bei 125 kBit/s. Eine zusätzliche Eigenschaft ist die Möglichkeit der Datenübertragung über nur eine Busleitung bei Ausfall einer Leitung. Selbst bei einem Kurzschluss zwischen den beiden Busleitungen ist noch eine Datenübertragung möglich. Aus diesem Grund wird die Zweidrahtausführung des Low-Speed-CAN auch als fault tolerant (dt. fehlertolerant) bezeichnet und mit FT-CAN abgekürzt.

Allerdings verliert der FT-CAN immer mehr an Bedeutung in der Kraftfahrzeugelektronik. Dies liegt hauptsächlich daran, dass der CAN-Bus sich immer mehr zum Standard-Bus entwickelt hat und die Kosten für High-Speed-CAN-Controller damit günstiger werden. Dazu kommen in das Kraftfahrzeug immer mehr Bussysteme, die für spezielle Anforderungen entwickelt wurden (z.B. LIN, MOST, FlexRay). Die Automobilhersteller versuchen diese Vielzahl an verschiedenen Bussystemen in Grenzen zu halten. Daher werden die Steuergeräte, die über Low-Speed-CAN miteinander kommunizieren, zukünftig entweder als intelligente Sensoren/Aktoren über LIN oder gleich über High-Speed-CAN vernetzt werden.

Auch sein Alleinstellungsmerkmal gegenüber anderen Datenbussen, die Fehlertoleranz, kann daran nichts ändern. Der FT-CAN wird auf Grund seiner geringen Datenübertragungsrate nicht als Sicherheitsbus eingesetzt. Sein Einsatzfeld ist der Komfortbereich, dort arbeitet er üblicherweise mit einer Übertragungsrate von 100 kBit/s. Im Fehlerfall signalisiert der FT-CAN einen Fehler, der behoben werden muss. Die Datenübertragung funktioniert auf Grund der Fehlertoleranz weiter. Bei einer Vernetzung mit dem High-Speed-CAN würde der Fehler zum Ausfall von Systemen/Funktionen führen. Da diese im Komfortbereich unkritisch sind, wird der Fehler nach dem Ausfall behoben. In beiden Fällen muss eine Reparatur erfolgen. Die Kostenersparnisse durch den Verzicht auf ein weiteres Datenbussystem überwiegen aus Sicht des Automobilherstellers die Komfortvorteile für den Kunden bei weitem.

3.3.5.2 High-Speed (HS-CAN)

Als der wichtigste Datenbus bei der Vernetzung von Kraftfahrzeugen gilt der High-Speed-CAN, auch als HS-CAN abgekürzt. Der HS-CAN nutzt den gleichen Data Link Layer (ISO 11898-1) wie der FT-CAN. Der Physical Layer des HS-CAN ist in der ISO 11898-2 standardisiert. Er setzt eine Linientopologie voraus und muss im Gegensatz zum FT-CAN an seinen jeweiligen Busenden terminiert sein. Die maximale Datenübertragungsrate beträgt beim HS-CAN 1 MBit/s. Dadurch wird allerdings die maximale Buslänge auf 40 m beschränkt. Da Busteilnehmer über Stichleitungen an den CAN-Bus angeschlossen sind, die Reflexionen verursachen, wird der HS-CAN im Kraftfahrzeug mit einer Übertragungsrate von 500 kBit/s betrieben. Die maximale Buslänge verlängert sich damit auf 100 m.

Zur Übertragung „normaler“ Prozessdaten wie Geschwindigkeit, Drehzahl usw. ist der HS-CAN auch zukünftig geeignet. Allerdings sind die 500 kBit/s des HS-CAN für die wachsenden Datenmengen, die in modernen Kraftfahrzeugen übertragen werden müssen, nicht mehr ausreichend. Durch Einsatz domänenspezifischer CAN-Busse wird versucht, diese Datenflut zu entkoppeln und auf mehrere CAN-Busse zu verteilen. Auf lange Sicht ist davon auszugehen, dass der HS-CAN durch Datenbusse höherer Übertragungskapazität wie beispielsweise FlexRay ergänzt wird.

3.3.5.3 Eigenschaften des CAN-Protokolls

Das CAN-Protokoll basiert nicht auf einem Datenaustausch durch Adressierung des Nachrichtenempfängers, sondern erfolgt durch Kennzeichnung einer übertragenen Nachricht über eine Nachrichtenerkennung (Nachrichten-Identifier). Alle Steuer-

geräte (Netzknoten) prüfen anhand der Kennung einer empfangenen Nachricht, ob diese für sie relevant ist. Nachrichten können daher von keinem, einem, vielen oder allen Teilnehmern übernommen werden (Multicasting, Broadcasting). Über die in den Protokoll-Controllern integrierten Möglichkeiten der Nachrichtenfilterung können Teilnehmer von der Auswertung nicht interessierender Nachrichten entlastet werden.

Da der Identifier einer Nachricht gleichzeitig dessen Priorität in Bezug auf den Buszugriff bestimmt, ist es möglich, Nachrichten entsprechend ihrer Wichtigkeit einen entsprechend schnellen Buszugriff zu ermöglichen. Besonders wichtige Nachrichten können damit unabhängig von der augenblicklichen Busbelastung den Zugang zum Bus mit kurzer Latenzzeit (Verzögerung) erlangen.

Die maximale Datenlänge einer CAN-Nachricht ist auf 8 Byte begrenzt. Mit dieser Datenlänge können die im Kfz-Bereich anfallenden Informationen in den meisten Fällen mit einer CAN-Nachricht übertragen werden. In diesen Anwendungsbereichen besteht die Anforderung weniger in der Übertragung langer Datenblöcke als vielmehr in der Realisierung einer möglichst hohen Nachrichtenrate bei kurzer Nachrichtenlänge. Die Begrenzung der Nachrichtenlänge auf wenige Bytes ist andererseits erforderlich, um eine möglichst kurze Latenzzeit für den Buszugriff hochpriorer Nachrichten zu gewährleisten. Für ein CAN-Netzwerk mit 8 Byte langen Nachrichten beträgt die maximal mögliche Latenzzeit für die höchstpriore Nachricht 130 Bitzeiten (entspricht 260 µs bei 500 kBit/s). Diese kurzen Latenzzeiten von hochprioren Nachrichten ermöglichen die Quasi-Echtzeitfähigkeit über den CAN-Bus, wie sie beim Antriebsstrang gefordert ist, bieten allerdings keinen Determinismus.

3.3.6 Local Interconnect Network (LIN)

Die Zunahme der Rechenleistung von Mikrocontrollern bei gleichzeitiger Miniaturisierung und sinkenden Preisen ermöglichen die Entwicklung von immer leistungsfähigeren Sensoren und Aktoren. Diese sind damit in der Lage, eine Vorverarbeitung von Daten vorzunehmen, die zu übertragende Datenmenge zu reduzieren und sie über ein Bussystem fahrzeugweit zur Verfügung zu stellen. Allerdings macht es aus Kosten- und Aufwandsgründen wenig Sinn, die Vernetzung über den CAN-Bus zu realisieren. Viele Automobilhersteller entwickelten für diese Anforderungen eigene proprietäre Lösungen. Der Local Interconnect Network-Bus (LIN) soll als kostengünstiger Standardbus diese Insellösungen der Hersteller ersetzen. Im Jahr 1998 wurde das LIN-Konsortium gegründet und hat sich zur Aufgabe gemacht, die Kosten und den Aufwand des Kommunikationsmanagement zu reduzieren [LIN]. Local Interconnect bedeutet, dass sich alle Steuergeräte innerhalb eines begrenzten Bauraums (z. B. Dach) befinden. Dieses wird auch als „lokales Subsystem“ bezeichnet. Die Abbildung 23 zeigt eine Einsatzmöglichkeit von LIN als „Spiegelbus“. Die Vorteile sind die Einsparungen an Kabeln, Steuergeräten und Kontakten sowie die Entkopplung von der Türverriegelung.

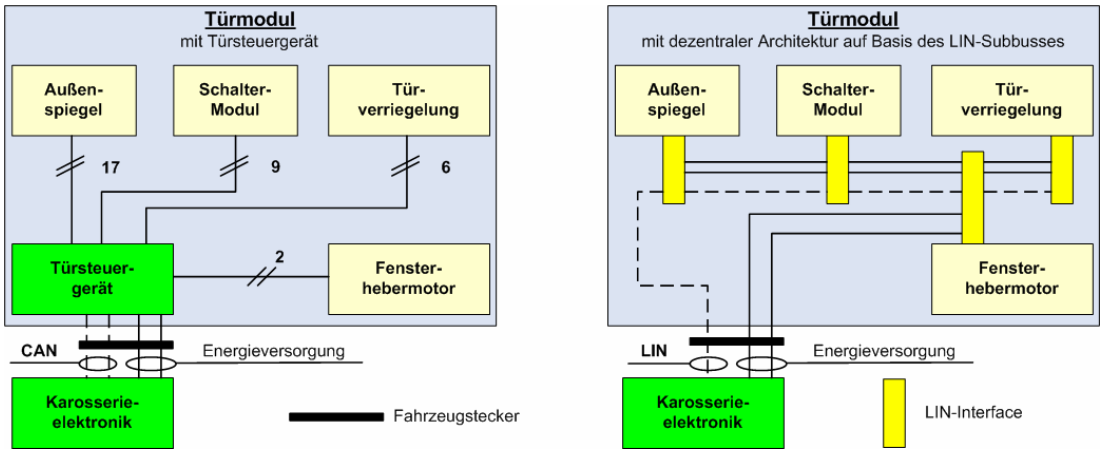

Abbildung 23: Einsatzmöglichkeit des LIN-Bus [Grzemba04]

Im September 2003 wurde die derzeit gültige LIN-Spezifikation in der Version 2.0 veröffentlicht. Der LIN-Bus ist ein Eindrahtbus, dessen Physical Layer auf der Norm ISO 9141 basiert. Damit werden Übertragungsraten bis 19,2 kBit/s ermöglicht. Bei LIN handelt es sich um eine Master-Slave-Architektur und als Zugriffsverfahren kommt das Delegated-Token-Prinzip zum Einsatz. Die Masterfunktion wird in der Regel vom CAN-Steuergerät übernommen und es können bis zu 16 LIN-Slaves an einem LIN-Bus angeschlossen sein. Die maximale Nachrichtenlänge auf dem LIN-Bus beträgt 8 Bytes. Ähnlich wie CAN ist LIN nachrichtenorientiert und kann bis zu 64 Nachrichten adressieren, von denen einige für spezielle Inhalte wie beispielsweise Diagnose reserviert sind. Weiterführende Informationen können der LIN-Spezifikation [LIN04] und der Dokumentation des LIN-Aufbaus entnommen werden, der im Rahmen dieser Arbeit entwickelt wurde [SchadeF04].

Mit der LIN-Spezifikation entstand in sehr kurzer Zeit ein Standard, der sehr schnell von der Automobilindustrie angenommen wurde. Bereits im Jahr 2003 und damit schon 5 Jahre nach Gründung des Konsortiums befand sich der LIN-Bus im Serieneinsatz (z.B. Audi A8). Der LIN-Bus unterstützt mit seinen Eigenschaften sehr kostengünstig die verteilten Systeme von intelligenten Sensoren/Aktoren und ist damit für mechatronische Steuerungen im Fahrzeug prädistiniert. Er wird auch in zukünftigen Kraftfahrzeugen eine Rolle spielen.

3.3.7 Media-Oriented Systems Transport (MOST)

Der Begriff „Media Oriented Systems Transport“ (MOST) steht für ein Netzwerk mit medienorientiertem Datentransport. Im Gegensatz zum CAN-Datenbus werden adressorientierte Botschaften an einen bestimmten Empfänger übermittelt. Der MOST-Datenbus ist durch ein Konsortium aus Automobilherstellern, Zulieferern und Softwareherstellern (gegründet 1998) entwickelt wurden, um multimediale Inhalte (Medien) im Kraftfahrzeug zu transportieren [MOST]. Eine besonders wichtige Rolle für den Transport solcher Infotainment-Inhalte (**Info**rmation und Enter**tainment**) spielt eine hohe Datenübertragungsrate, die bei MOST maximal 24,8 MBit/s beträgt. Über eine Ringtopologie können bis zu 64 Geräte miteinander über einen MOST-Bus vernetzt werden. Den Physical Layer spezifiziert MOST optisch für Lichtwellenleiter (LWL), deren Lichtwellen mit einer Wellenlänge von

650 nm als rotes Licht sichtbar sind. Dadurch werden die hohe Datenraten und die hohe Störsicherheit ermöglicht. Nachteile der LWL's sind die aufwendigen mechanischen Steckverbindungen, die Empfindlichkeit gegenüber hohen Temperaturen und die eingeschränkten Möglichkeiten bei der Verlegung (z.B. Biegeradius > 2,5 cm). Daher arbeitet derzeit das MOST-Konsortium an einer Spezifikation für einen elektrischen Physical Layer.

MOST versendet Botschaften in einer festen Taktfrequenz von 44,1 kHz, was der Übertragungsfrequenz digitaler Audiodaten entspricht. Eine Botschaft hat eine Länge von 64 Byte, wobei je 1 Byte für Header und Trailer sowie 2 Bytes für Steuerungsdaten genutzt werden. Die restlichen 60 Bytes können zwischen den synchronen Daten (24 bis 60 Bytes) und asynchronen Daten (0 bis 36 Bytes) aufgeteilt werden. Mit diesem Nachrichtenformat lassen sich maximale Übertragungsraten von 24,8 MBit/s für synchrone Daten, 14,4 MBit/s für asynchrone Daten und garantierte 700 kBit/s für Steuerungsdaten erzielen.

Durch seine Konzeption eignet sich MOST besonders für Multimediadaten. Aufgrund seiner Kosten und seines Aufwandes sowie seines optischen Übertragungsmediums wird MOST auf den Bereich des Infotainments beschränkt bleiben. Seine hohe Übertragungsrate macht ihn allerdings für Spezialanwendung wie beispielsweise das Flashen interessant. Mit einem Diagnosezugang direkt zum MOST wäre eine Update-Programmierung mit mehr als der zehnfachen Geschwindigkeit, wie sie CAN idealerweise bietet, möglich. Einige Fahrzeughersteller nutzen den MOST auch bei ausgewählten Fahrzeugmodellen für die Update-Programmierung (z.B. BMW). Grundsätzlich ist der optische Bus aber für die raue Kundendienstumgebung und das damit verbundene häufige An- und Abstecken ungeeignet.

Durch seine Spezialisierung auf den Multimedia-Bereich eines Kraftfahrzeugs bleibt die Verbreitung des MOST begrenzt, mit den Folgen der vergleichsweise hohen Systemkosten. Dazu kommen zunehmendes Datenaufkommen durch den Einsatz zusätzlicher Multimedia-Technik und neuer Technologien, bei denen der MOST-Bus an seine Grenzen stößt. Aus diesem Grund wird derzeit an der zweiten Generation des MOST gearbeitet, die eine Datenrate von 50 bis 150 MBit/s unterstützt und sowohl einen optischen als auch einen elektrischen Physical Layer bieten soll. Mit der neuen Spezifikation existieren auch Bestrebungen, den MOST in der Unterhaltungselektronik zu etablieren. Dies würde die Verbreitung erleichtern, die Kosten senken und Synergieeffekte zwischen Unterhaltungselektroniksystemen und Kraftfahrzeug ermöglichen. Genau aus diesem Bereich erwächst dem MOST eine starke Konkurrenz. Mit dem IDB 1394 (Kapitel 3.3.9.1) kommt ein etablierter Datenbus aus der IT- und Unterhaltungselektronik, der flexibler als der derzeitige MOST ist (elektrisch, optisch, Topologie usw.), um ein Vielfaches schneller ist und eine leichte Integration bestehender Consumer-Elektronik in das Kraftfahrzeug ermöglicht.

Eine genaue Prognose bezüglich des Infotainmentbusses zukünftiger Generationen kann derzeit nicht getroffen. Der MOST in seiner jetzigen Form ist etabliert und wird von allen Fahrzeugherstellern unterstützt. Zukünftig wird sich in diesem Bereich aber erst zeigen müssen, welches Konzept, MOST der zweiten Generation oder IDB, die Fahrzeughersteller bevorzugen werden.

3.3.8 FlexRay

Mit der Zunahme zeitgesteuerter Systeme im Fahrzeug und neuen Funktionen wie beispielsweise X-by-Wire wächst auch der Bedarf an zeitgesteuerten und deterministischen Kommunikationsbussystemen. Die Automobilindustrie hat dies erkannt und Ende der neunziger Jahre mit Entwicklungen für solch ein zeitgesteuertes Kommunikationssystem begonnen. Erste Erfolge wurden mit TTP (Kapitel 3.3.9.4) erzielt bis sich Anfang dieses Jahrzehnts die Automobilindustrie zugunsten eines einheitlichen Standards auf FlexRay als zeitgesteuertes Kommunikationssystem geeinigt hat. Dem FlexRay Konsortium gehören mittlerweile über 100 Firmen aus der Automobil-, Zulieferer-, Software- und Halbleiterindustrie an [FlexRay].

Der FlexRay Standard definiert einen zweikanaligen Datenbus über verdrillte Kupferkabel wie CAN oder optische LWL. Pro Kanal stellt FlexRay eine Datenrate von 10 MBit/s zur Verfügung. Die Flexibilität war Grundanforderung bei der Entwicklung von FlexRay und gab dem Datenbus seinen Namen. Die Topologie kann je nach Anwendung frei zwischen Linien-, Stern- und kaskadierten Sternen sowie Mischformen davon gewählt werden. Bei Einsatz des FlexRay als Sicherheitsbus werden die Informationen auf beiden Kanälen redundant übertragen. In nicht sicherheitskritischen Anwendungen kann der optionale Übertragungskanal zur Bandbreitenerhöhung genutzt werden.

Die Datenübertragung bei FlexRay geschieht in Kommunikationszyklen. Diese Kommunikationszyklen können aus einem statischen (zeitgesteuerten) und einem dynamischen (quasi ereignisgesteuerten) Teil bestehen. Das Verhältnis zwischen statischen und dynamischen Teil bestimmt der Anwender je nach Anforderungen. Der statische Teil wird in eine Anzahl von Zeitschlitzen gleicher Größe unterteilt, in denen zyklisch Daten ausgetauscht werden. Im dynamischen Teil werden nur Daten übertragen, wenn es erforderlich ist. Der Buszugriff wird dabei über ein Minislotting-Verfahren mittels Prioritätenvergabe geregelt [FlexRay05]. Einzelheiten zur Datenübertragung bei FlexRay werden im Kapitel 5.2.1 bei der Realisierung des Kommunikationsnetzwerkes beschrieben.

FlexRay lebt von einer breiten Unterstützung durch die Automobilindustrie und wird dort als Nachfolger für den CAN gehandelt. Durch BMW soll 2006 der FlexRay erstmals in Serie seine Leistungsfähigkeit unter Beweis stellen. Es gilt als sicher, dass FlexRay zukünftig einen ähnlichen Stellenwert wie derzeit CAN einnimmt bekommen wird. Mit der zweiten Generation, die eine Datenübertragungsrate von 100 MBit/s besitzen soll, ist FlexRay auch für die Zukunft gerüstet. Trotz alledem ist davon auszugehen, dass FlexRay den CAN nicht ersetzen, sondern ergänzen wird. Sein Einsatz wird dabei nicht auf X-by-Wire-Anwendungen beschränkt bleiben. Durch seine Eigenschaften und seine zehnmal höhere Bandbreite als CAN eignet er sich beispielsweise als Backbone (siehe auch Kapitel 3.4), wie es im Rahmen dieser Arbeit realisiert wurde.

3.3.9 Weitere automobile Datenbusse

Im Folgenden werden Datenbusse vorgestellt, die noch keine große Bedeutung oder keine Bedeutung mehr für die Automobilindustrie haben. Sie werden der

Vollständigkeit halber erwähnt, da zukünftige Datenbusarchitekturen zu bestehenden Bussystemen kompatibel und neue fortschrittliche Bussysteme leicht integrierbar sein müssen.

3.3.9.1 Intelligent Transportation System Data Bus (IDB)

Mit dem Intelligent Transportation System Data Bus (IDB) wurde für den in der Unterhaltungselektronik sehr erfolgreichen Datenbus IEEE 1394 (auch bekannt als Firewire oder i-Link) eine Automobilversion entwickelt [IDB]. Der verbreitete Standard IEEE 1394a spezifiziert Datenübertragungsraten von 100, 200 und 400 MBit/s. Der aktuelle Standard IEEE 1394b erweitert diese Datenraten um 800, 1600 und 3200 MBit/s. Als Physical Layer können elektrische und optische Übertragungsmedien zum Einsatz kommen. Das Konzept des IDB verfolgt den Einsatz von Consumer-Geräten (Notebooks, PDAs, Navigationsgeräte, DVD-/CD-Player, Kameras usw.) im Fahrzeug. Diese können über einen Customer-Convenience-Port (CCP), der Bestandteil der IDB-Spezifikation ist, an das Fahrzeugnetzwerk angeschlossen werden. Die Einführung des IDB ist für Anfang 2006 geplant.

IEEE 1394 besitzt eine isochrone und eine asynchrone Übertragung. Die isochrone Übertragung belegt 80 % der Bandbreite und arbeitet ohne Fehlerkorrektur. Der Rest steht für die asynchrone Übertragung zur Verfügung, die eine Fehlerkorrektur besitzt. Daten mit „harten“ Realzeitbedingungen lassen sich mit IEEE 1394 nicht bzw. nicht optimal transferieren [Huber04]. Doch seine hohen Datenraten machen den IEEE 1394 auch für andere Anwendungen innerhalb des Kraftfahrzeugs interessant. In der Automatisierungstechnik werden Konzepte umgesetzt, die sich zum Ziel gesetzt haben, die Bandbreitenreserven aufgrund der hohen Datenrate für eine Echtzeitfähigkeit zu nutzen [Beckmann02].

3.3.9.2 Ethernet

Sehr hohe Datenübertragungsraten bietet Ethernet mit derzeit bis zu 1 GBit/s. Dazu kommt die sehr starke Verbreitung innerhalb der IT-Industrie, was auf den offenen Standard zurückzuführen ist und kostengünstige Bauteile verspricht. Es existieren ebenfalls Bestrebungen nach Ethernet-Echtzeitversionen mit Industrial Ethernet oder ProfiNet. Trotz allem konnte sich Ethernet bislang im Fahrzeug nicht durchsetzen. Aufgrund seiner Eigenschaften wird Ethernet aber auch in Zukunft für die Automobilindustrie interessant bleiben und ein Einsatz im Kraftfahrzeug diskutiert werden.

3.3.9.3 Automotive Safety Restraints Bus (ASRB) – Safe by Wire

Alle der derzeit existierenden Datenbusse für sicherheitskritische Anwendungen wie FlexRay, TTP/C oder byteflight haben den entscheidenden Nachteil, dass sie sehr aufwändig und sehr teuer sind. Ein Konsortium aus Automobilzulieferern hat sich in der Safe-by-Wire-Gruppe aus diesem Grund zusammengeschlossen, einen Sensor/Aktorbus (ähnlich zum LIN) für sicherheitskritische Anwendungen wie beispielsweise für eine Airbagsteuerung zu entwickeln. Die Spezifikation des Automotive Safety Restraints Bus (ASRB) liegt in der Version 2.0 vor und soll als ISO 22896 standardisiert werden.

Der Zweidrahtbus unterstützt eine bidirektionale Master-Slave-Kommunikation, bei der sich die Slaves über den Bus mit Energie versorgen. Die flexible Bustopologie erreicht Datenraten von 20, 40, 80 oder 160 kBit/s, die vom Master vorgegeben und während des Betriebs geändert werden können. Erstes Silizium ist für diesen Bus schon verfügbar. Eine Aussage über die zukünftige Bedeutung für die Fahrzeugvernetzung kann zu diesem Zeitpunkt allerdings noch nicht getroffen werden.

3.3.9.4 Time Triggered Protocol (TTP /A, TTP/C)

Das Time Triggered Protocol entstand an der TU Wien in Rahmen von Forschungen zu Echtzeitsystemen. Es ist für sicherheitskritische Anwendungen mit harten Echtzeitforderungen konzipiert [TTP]. Diese Eigenschaften machten TTP für die Automobilindustrie interessant, die maßgeblich an der Entwicklung mitgewirkt hat. Allerdings führten die unflexible und statische Datenübertragung von TTP zur Entwicklung von FlexRay. Alle großen Automobilhersteller haben sich für FlexRay entschieden, so dass TTP in der Automobilindustrie keine Rolle mehr spielt. In der Flugzeugindustrie und Automatisierungstechnik findet TTP seine Anwendung.

Es existieren mehrere Ausführungen von TTP, die entsprechend den SAE-Klassen (siehe Kapitel 3.3.3.5) eingeordnet sind. Am unteren Ende der Leistungsfähigkeit steht der TTP/A, der als Eindrahtbus über keine Sicherheits- und Echtzeitfähigkeiten verfügt. Er kann maximal 20 kBit/s übertragen und steht in direkter Konkurrenz zum LIN. Da ein TTP/A-Knoten ungefähr doppelt so teuer ist wie ein vergleichbarer LIN-Knoten, ist TTP/A im Fahrzeug nicht relevant.

Am oberen Ende der Leistungsfähigkeit mit maximal 25 MBit/s Übertragungsrate (optisch) steht der TTP/C. Dieser Datenbus erfüllt Sicherheits- und Echtzeitanforderungen. Seine Mechanismen zur Datenübertragung sind denen vom statischen Teil des FlexRay ähnlich. Der wesentliche Unterschied besteht in der variablen Slotlänge des TTP/C für die einzelnen Knoten, während FlexRay eine konstante Slotlänge für alle Knoten vorschreibt.

3.3.9.5 Time Triggered CAN (TT-CAN)

Da die nachrichtenorientierte Adressierung bei CAN für harte Echtzeitanforderungen nicht ausreicht, wurde Time Triggered CAN (ISO 11898-4) entwickelt, das eine Zeitsteuerung auf dem CAN-Bus aufsetzt [CAN]. Dazu werden vom Anwender Zeitslots definiert, die jeweils einem Master zur Verfügung stehen, bzw. in denen auch mehrere Master nach dem CAN-Arbitrationsprinzip Nachrichten übertragen können. So lassen sich Echtzeitbedingungen und maximale Datentransferrate gegeneinander abwiegen. Zusätzlich existiert ein Zeitmaster innerhalb eines CAN-Clusters, der in regelmäßigen Zeitabständen Synchronisationsframes sendet. Dieser zeitgesteuerte Ansatz auf dem CAN erzeugt zusätzlichen Overhead für die mit 1 MBit/s schon relativ begrenzte Bandbreite des CAN. Daher wird TT-CAN im Fahrzeug zukünftig keine Rolle spielen.

3.3.9.6 Byteflight

Dieser Datenbus wurde von BMW (1999) als Sicherheitsbus entwickelt und übernimmt dort die Vernetzung der Crashsensoren, Airbags und Auslösungssteuerung im, von BMW entwickelten, Intelligent-Safety-Integration-System (ISIS). Das Protokoll spezifiziert weder Topologie noch Physical Layer. BMW setzt byteflight als aktiven Stern mit einer optischen Verkabelung mittels LWL ein. Die Übertragungsrate von byteflight beträgt 10 MBit/s. Zugunsten von FlexRay wird byteflight allerdings nicht mehr weiterentwickelt und bei BMW mittelfristig durch FlexRay ersetzt werden. Ein großer Teil der byteflight-Spezifikation ist in den FlexRay Standard als dynamischer Teil des FlexRay eingeflossen.

3.4 Drahtlose Datenübertragungsprotokolle

Neben den drahtgebundenen Datenübertragungsverfahren existiert eine Reihe von Funktechnologien zur drahtlosen Datenübertragung. Da durch den Wegfall des Kabels Gewicht, (Material-)Kosten und Verkabelungsaufwand eingespart werden, sind die drahtlosen Datenübertragungsverfahren nicht nur für die Automobilindustrie sehr interessant. Dazu kommt, dass in den letzen Jahren große Fortschritte auf dem Gebiet der drahtlosen Übertragungen erzielt wurden. Immer mehr Funktechnologien sind für die verschiedenen Anwendungen und Einsatzgebiete entstanden. Das Hauptanwendungsgebiet für drahtlose Übertragungen im Kraftfahrzeugbereich ist die Telematik. Der Begriff „Telematik“ setzt sich aus den Wörtern **Tele**kommunikation und Infor**matik** zusammen und befasst sich mit dem Transport und der Verarbeitung von verkehrs- und fahrzeugbezogenen Informationen.

Allerdings bleiben die Einsatzfelder für drahtlose Übertragungsverfahren aufgrund deren Eigenschaften im Fahrzeug sehr begrenzt. Im Gegensatz zu drahtgebundenen Verfahren fehlt ihnen die Exklusivität des Übertragungsmediums. Es müssen mit konkurrierenden Übertragungsverfahren im gleichen Frequenzbereich, Störungen und Signaldämpfungen durch Hindernisse im Übertragungsweg gerechnet werden. Die Bandbreite der einzelnen drahtlosen Verfahren muss dabei immer zwischen der Anzahl der Teilnehmer, die gleichzeitig kommunizieren möchten, geteilt werden. Dadurch ist die Datenübertragungsrate starken Schwankungen unterworfen und kann unter Umständen bis zum Übertragungsabbruch führen. Auf Grund dieser unvorsagbaren Verfügbarkeit kommen drahtlose Übertragungsverfahren für sicherheitskritische Anwendungen nicht in die engere Auswahl. Darüber hinaus bieten sie durch die Nichtexklusivität des Übertragungsmediums von sich aus keine Sicherheit gegenüber Abhören und Datenmanipulationen. Diese Sicherheit kann nur durch zusätzliche Mechanismen in den oberen Protokollschichten des jeweiligen Übertragungsprotokolls geschaffen werden. Die verschiedenen drahtlosen Übertragungsprotokolle betreiben dabei unterschiedlichen Aufwand für die Übertragungssicherheit.

Drahtlose Datenübertragungsverfahren nutzen modulierte elektromagnetische (Funk-)Wellen zur Übertragung von Daten. Gerade in einem EMV-sensitiven Umfeld wie in einem Automobil ist ein Einsatz deshalb problematisch. Die Risiken liegen in Störungseinkopplungen auf drahtgebundene Datenbusse oder Fehlfunktionen durch Störungen. Als extremes Beispiel sei hier die Fehlauslösung des

Airbags durch elektromagnetische Störungen genannt. Es muss im Vorfeld gewährleistet sein, dass keine Systemfunktionen beeinträchtigt werden. Diese Forderung setzt intensive und aufwändige EMV-Tests voraus.

Die drahtlose Übertragung geschieht durch das Senden und Empfangen elektromagnetischer Wellen. Üblicherweise wird auf eine konstante hochfrequente Trägerfrequenz vom Sender nach bestimmten Methoden (z.B. phasen- oder frequenzmoduliert usw.) die Nachricht als variables niederfrequentes Signal aufmoduliert und vom Empfänger anschließend wieder demoduliert. Die Trägerfrequenz bestimmt im Wesentlichen die Übertragungseigenschaften der Funksignale. Die Frequenzen sind in einzelne Frequenzbereiche unterteilt, für die nationale Regelungen bezüglich Sendeleistung, Störstrahlung, Frequenzbreite usw. existieren. Ein großer Teil dieser Frequenzbereiche ist darüber hinaus für bestimmte Anwendungen reserviert oder muss mit hohen Kosten lizenziert werden. Aus diesem Grund ist das ISM-Band (**I**ndustrial, **S**cientific and **M**edical-Band) interessant, das nicht der staatlichen Regulierung unterliegt und lizenzfrei genutzt werden kann. Einzige Auflagen betreffen die maximalen Sendeleistungen und die Störungen benachbarter Frequenzbereiche.

Frequenzbereich des ISM-Band	**Bemerkung**
13.553 kHz – 13.567 kHz	Smart-Tags (RFID)
26.957 kHz – 27.283 kHz	Modellbau-Fernsteuerungen, CB-Funk
40,66 MHz – 40,70 MHz	Modellbau-Fernsteuerungen
433,05 MHz – 434,79 MHz	Sehr stark genutzt, Überschneidung mit Amateurfunk
868,0 MHz – 870,0 MHz	Besondere Vorschriften für diesen Bereich
2,4 GHz – 2,5 GHz	Sehr stark genutzt (W-LAN, Bluetooth)
5,125 GHz – 5,875 GHz	W-LAN, HiperLAN
24,0 GHz – 24,25 GHz	Nahbereichsradar für PKW (ab 2013 bei 77 GHz)

Tabelle 2: Ausgewählte Frequenzbereiche des ISM-Bandes zur drahtlosen Kommunikation

Das ISM-Band teilt sich in mehrere Frequenzbereiche auf, wovon die wichtigsten für drahtlose Übertragungen in Tabelle 2 aufgelistet sind. Allerdings sind diese weltweit nicht einheitlich geregelt. In den USA gibt es beispielsweise drei ISM-Bänder (902 – 928 MHz, 2.400 – 2.483,5 MHz und 5.800 MHz). Das 2,4-GHz-Band ist als einziges weltweit freigegeben, während die Freigabe des 5-GHz-Bandes für diesen Zweck noch diskutiert wird. Der große Vorteil des ISM-Bandes, seine kostenfreie Nutzung für alle Anwendungen, ist auch gleichzeitig der große Nachteil. Insbesondere die für drahtlose Übertragung gut geeigneten Frequenzbereiche von 433 MHz bis 2,4 GHz sind mittlerweile sehr stark überlastet.

3.4.1 Technische Anwendungen drahtloser Übertragungstechnik bei Kraftfahrzeugen

Eine der ersten Anwendungen drahtloser unidirektionaler Übertragung im Kraftfahrzeugbereich und lange Zeit auch die einzige stellt der unkritische Empfang von Radiowellen dar. Die Antenne für die Lang-, Mittel-, Kurz- und Ultrakurzwellenprogramme (LW, MW, KW, UKW) befindet sich außerhalb des Fahrzeugs. Die dort eingekoppelten Signale werden per Kabel zum Radio übertragen. An diesem Prinzip hat sich bis heute nichts geändert, unabhängig von der analogen oder digitalen Übertragung der Radiosender. Neben der Übertragung von „Inhalt" (Radiosendern) werden Zusatzinformationen wie Text- und Verkehrsmeldungen (RDS, TMC) gesendet. Die digitalen Verfahren schaffen durch ihre Übertragungstechnik im Vergleich zur analogen Übertragung hohe Datenraten (DAB 1,8 MBit/s, DVB-T 35 MBit/s), die für zusätzliche Dienste genutzt werden können.

Zum Anfang der neunziger Jahre wurden bei den meisten Fahrzeugherstellern ein drahtloses Übertragungsverfahren zur automatischen Öffnung und Verriegelung des Fahrzeugs eingeführt. Dieses Verfahren sendete im Infrarotbereich, funktionierte nur auf sehr kurze Entfernungen und benötigte „Sichtkontakt" zum Fahrzeug. Gegen Mitte der neunziger Jahre stellte man das relativ „unkomfortable" Übertragungsverfahren auf Funkwellen um. Damit erhöhte sich die Reichweite der Schlüssel auf mindestens 10 Meter. Im Fahrzeugschlüssel ist dazu ein RFID-Chip[17] untergebracht, auf dem kodierte Daten zur Zugangskontrolle gespeichert sind. Per Funk wird die Bordelektronik des Fahrzeugs „geweckt" und kommuniziert mit dem RFID-Chip über den Funkkanal. Der liegt dabei meist im ISM-Band bei 433 MHz. Während dieser Übertragung werden diverse Schlüsselcodes zur Identifikation ausgetauscht und abgeglichen. Bei neueren Systemen (z.B. „Easy Entry") wird ab einer bestimmten minimalen Entfernung des Fahrzeugschlüssels vom Fahrzeug gleichzeitig die Wegfahrsperre deaktiviert.

Herkömmliche Wegfahrsperren ohne „Easy Entry" nutzen ebenfalls eine drahtlose Übertragungstechnik. Hier kommt häufig eine passive Transpondertechnik zum Einsatz, die eine Trägerfrequenz von 125 kHz nutzt und nur auf sehr kurze Reichweiten funktioniert. Der Fahrzeugschlüssel muss dazu im Zündschloss stecken. Sein Transponder (RFID-Chip) wird im Zündschloss gleichzeitig über das Funksignal mit Energie versorgt. Während dieser Kommunikation werden ebenfalls Schlüsselcodes übertragen.

Die elektronische Reifendrucküberwachung ist eine weitere technische Anwendung drahtloser Übertragungstechnik im Kraftfahrzeug. Dieses Sicherheits-Sensorsystem ist in den USA Pflicht für neu zugelassene Fahrzeuge und wird auch in Europa immer häufiger angeboten. Die eingesetzten Verfahren zur Messung und Signalübertragung sind unterschiedlich. Die Sensoren, die aktiv (mit Batterie) oder passiv sein können, befinden sich entweder auf der Felge oder sind direkt in die Reifen einvulkanisiert. Derzeitige Sensoren messen dabei den Reifendruck und die Reifentemperatur. Die Vision der Zukunft für dieses Sensorsystem sind „intelligente" Reifen, die gleichzeitig Quer- und Längskräfte messen können, mit

[17] Radio Frequency Identification (RFID) (engl. für Funk-Erkennung) ist eine Methode, um Daten auf einem Transponder berührungslos und ohne Sichtkontakt lesen und speichern zu können.

denen eine Online-Reibwertschätzung möglich wäre. Die Übertragungsverfahren der Sensoren beruhen auf proprietären Funkprotokollen im Frequenzbereich von 125 kHz oder im ISM-Band von 2,4 GHz.

Diese Anwendungen haben eine relativ geringe Datenübertragungsrate gemeinsam, die mit den dabei verwendeten Übertragungsprotokollen und –technologien erreicht werden. Für größere Datenmengen, wie sie beispielsweise die Update-Programmierung fordert, sind diese eingesetzten drahtlosen Übertragungstechniken ungeeignet. Aus diesem Grund werden in den nächsten Abschnitten für die Automobilindustrie interessante drahtlose Übertragungsprotokolle vorgestellt.

3.4.2 Global System for Mobile Communications (GSM)

Das Global System for Mobile Communications (GSM) ist ein volldigitaler Mobilfunknetz-Standard, der für Sprachübermittlung (Telefonie) aber auch für leitungsvermittelte und paketvermittelte Datenübertragung sowie Kurzmitteilungen (Short Messages Service - SMS) genutzt wird. Es ist der erste Standard der so genannten zweiten Generation (2G) von Mobilfunkstandards als Nachfolger der analogen Systeme der ersten Generation und der weltweit am meisten verbreitete Mobilfunk-Standard. GSM wurde mit dem Ziel geschaffen, ein mobiles Telefonsystem anzubieten, das Teilnehmern eine europaweite Mobilität erlaubt und mit ISDN oder herkömmlichen analogen Telefonnetzen kompatible Sprachdienste anbietet.

Im Kraftfahrzeugbereich wird GSM hauptsächlich für den Mobilfunk genutzt. Allerdings bietet GSM darüber hinaus die Möglichkeiten einer (groben) Ortung und der Datenübertragung, die es im Automobilbereich für einige andere Anwendungen interessant erscheinen lassen. So nutzen Telematiksysteme einiger Hersteller die SMS zur Signalisierung eines Notrufs mit Positionsinformationen. Die gleiche Information kann auch in Verbindung mit einem Diebstahlschutz zur Auffindung gestohlener Fahrzeuge genutzt werden. Mit Hilfe der Datenübertragung, die GSM mit 9,6 kBit/s[18] bietet, können kleinere Datenmengen bidirektional mit dem Fahrzeug ausgetauscht werden. Diese Möglichkeit nutzen beispielsweise einige Anbieter von Navigationssystemen für eine Offline-Navigation. Dabei wird das gewünschte Ziel dem Anbieter aus dem Fahrzeug übermittelt und die zur Zielführung notwendigen Daten in das Fahrzeug übertragen.

Das GSM-Verfahren wurde ursprünglich für Telefongespräche, Faxe und Datensendungen mit konstanter Datenrate konzipiert. Burstartige Datensendungen mit stark schwankender Datenrate, wie es beim Internet üblich ist, oder die asynchrone Übertragung hoher Datenmengen wurden nicht eingeplant. Mit dem Erfolg des Internets begannen Entwicklungen, bei der das GSM-Netz komplett abwärtskompatibel mit Möglichkeiten zur paketorientierten Datenübertragung in Verbindung mit einer Erhöhung der Datenübertragungsrate erweitert wurde.

[18] Eine fortschrittlichere Kanalkodierung ermöglicht 14,4 kBit/s, bewirkt aber bei schlechten Funkverhältnissen viele Blockfehler, so dass die Übertragungsrate niedriger ausfallen kann als es bei der Kodierung mit erhöhter Sicherheit auf dem Funkweg.

3.4.2.1 High Speed Circuit Switched Date (HSCSD)

Durch die Kopplung von mehreren Kanälen erreicht High Speed Circuit Switched Data (HSCSD) insgesamt eine höhere Datenrate von maximal 57,6 kBit/s. Allerdings handelt es sich bei HSCSD immer noch um ein verbindungsorientiertes Übertragungsverfahren, bei dem die Funkressource für die Dauer der Verbindung belegt bleibt, auch wenn keine Daten übertragen werden (wie bei z.B. Sprachverbindungen üblich).

3.4.2.2 General Packet Radio Service (GPRS)

Der General Packet Radio Service (GPRS) dagegen erlaubt zum ersten Mal eine paketvermittelte Datenübertragung. Dabei können Datenkanäle kurzfristig abgeschaltet und für andere Übertragungen genutzt werden, wenn aktuell keine Daten übertragen werden. Einzig schmalbandige Signalisierungskanäle bleiben bestehen, sodass die Datenkanäle bei Bedarf automatisch wieder geöffnet werden können. Der tatsächliche Datendurchsatz hängt unter anderem von der Netzlast ab und beträgt maximal 57,6 kBit/s[19]. Bei geringer Last kann ein Nutzer mehrere Zeitschlitze á 14,4 kBit/s parallel verwenden, während bei hoher Netzlast jeder GPRS-Zeitschlitz auch von mehreren Benutzern verwendet werden kann.

3.4.2.3 Enhanced Datarate for Global Evolution (EDGE)

Mit Enhanced Datarate for Global Evolution (EDGE) wurde durch eine neue Modulation (8PSK) eine Erhöhung der Datenrate ermöglicht. Mit EDGE werden GPRS zu EGPRS (Enhanced GPRS) und HSCSD zu ECSD (Enhanced Circuit Switched Data) erweitert. Bei EGPRS lassen sich je nach verwendeten Kodierungsverfahren pro Zeitschlitz Datenraten von 8,8 kBit/s bis zu 59,2 kBit/s erzielen. Damit würde EGPRS theoretisch bei voller Auslastung aller 8 Zeitschlitze eine maximale Datenübertragungsrate von 473,6 kBit/ ermöglichen. Wahrscheinlicher ist allerdings die Verwendung robusterer Kodierungsverfahren, die auch von UMTS benutzt werden und eine maximale Datenrate von 384 kBit/s bieten.

3.4.2.4 Nutzen für die Automobilindustrie

Für die Automobilhersteller ist der Einsatz von GSM zur Datenübertragung nicht sonderlich attraktiv. Interessant sind der Reifegrad des Standards und seine flächendeckende Verfügbarkeit (zumindest innerhalb Europas). Nachteilig dagegen sind die Kosten für GSM (Provider und Infrastruktur) und die relativ geringen Datenübertragungsraten. Selbst die weiter entwickelten Verfahren wie HSCSD, GPRS oder EDGE erreichen keine Datenübertragungsraten, die zur Update-Programmierung von einzelnen Steuergeräten ausreichen. Darüber hinaus müssten für solche Anwendungen weitere Randbedingen wie Kosten, Sicherheit und Wahrung der Datenintegrität diskutiert werden. Zudem steht mit dem neuen Mobil-

[19] Die Datenrate von 57,6 kBit/s entspricht der derzeit üblichen Bündelung von 4 Zeitschlitzen. Bei der maximalen Auslastung (8 Zeitschlitze) könnten theoretisch bis zu 115,2 kBit/s übertragen werden.

funkstandard UMTS der dritten Generation (3G) eine verbesserte Technologie bereit, mit der derartige Anwendungen in Betracht gezogen werden können.

3.4.3 Universal Mobile Telecommunications System (UMTS)

Bei dem Universal Mobile Telecommunications System (UMTS) handelt es sich um den Mobilfunkstandard der dritten Generation (3G), der das weit verbreitete GSM ablösen wird. Die Entwicklungsziele bei UMTS waren neben der Verbesserung der Sprachqualität bis hin zur Videotelefonie eine Erweiterung der multimedialen Dienste wie sie beispielsweise das Internet bietet. Grundvoraussetzung dafür ist eine signifikante Erhöhung der verfügbaren Bandbreite. UMTS bietet momentan eine Datenübertragungsrate von 384 kBit/s und nutzt dafür ein Frequenzmultiplex-verfahren. Zukünftige Techniken verwenden ein Zeitmultiplexverfahren und erreichen maximale Datenübertragungsraten von 1,92 MBit/s. Damit wären Anwendungen wie beispielsweise die Update-Programmierung von Fahrzeugen denkbar. Allerdings sind in diesem Fall eine Reihe von Fragen wie Sicherheit oder Geschäftsmodell zu klären. Der Update- und Programmierungs-Prozess muss bei den Fahrzeugherstellern vorhanden und erprobt sein, bevor Update-Aktionen im „Feld“ mit weit reichenden Konsequenzen für Kunden und Hersteller durchgeführt werden. Für die globale Vernetzung zukünftiger Fahrzeug bietet sich UMTS dagegen an.

3.4.4 IEEE 802.11 – Wireless Local Area Network (W-LAN)

Die am häufigsten eingesetzte drahtlose Übertragungstechnik im Bereich der lokalen Funknetze ist die Protokollfamilie des IEEE 802.11-Standards, allgemein als Wireless LAN[20] (W-LAN) bezeichnet. Ursprünglich mit einer Datenübertragungsrate von 2 MBit/s spezifiziert (IEEE 802.11), wurde W-LAN kontinuierlich weiter-entwickelt und in mehreren Versionen standardisiert. W-LAN sendet im ISM-Band je nach Version bei 2,4 GHz oder 5 GHz und erreicht maximale Datenübertragungs-raten bis zu 54 MBit/s. Einen Überblick über die verschiedenen IEEE 802.11-Standards gibt Tabelle 3.

Standard	**802.11**	**802.11b**	**802.11b+**	**802.11a**	**802.11g**
Frequenzbereich	2,4 GHz	2,4 GHz	2,4 GHz	5 GHz	2,4 GHz
Übertragungsrate	2 MBit/s	11 MBit/s	22 MBit/s	54 MBit/s	54 MBit/s
Kompatibel zu	802.11b	802.11b+/g	802.11b/g	–	802.11b/b+

Tabelle 3: Übersicht über die verschiedenen 802.11-Standards

Der Standard 802.11p, der in der Tabelle 3 nicht explizit aufgeführt ist, wird speziell für automobile Anwendungen entwickelt. Begonnen wurde damit im September 2004 und voraussichtlich 2007 soll der neue Standard 802.11p verabschiedet werden. Im Vordergrund stehen dabei die ad-hoc-Vernetzung von Fahrzeugen (Car-to-Car

[20] Im Prinzip stellt der IEEE 802.11-Standard nur eine Untermenge für W-LANs dar. Andere Übertragungsverfahren wie HiperLAN fallen ebenfalls unter W-LAN. Im Rahmen dieser Arbeit wird, wie im allgemeinen Sprachgebrauch üblich, W-LAN als Synonym für IEEE 802.11 verwendet.

Communication) und die Bildung dynamischer Netzwerke. Die Basis für dieses Automotive W-LAN ist der 801.11a-Standard, der um die parallele Nutzung mehrerer drahtloser Kanäle und der Festlegung eines Steuerungskanals erweitert wurde. Die Datenrate von 802.11p liegt mit diesen Änderungen zwischen 3 MBit/s und 27 MBit/s. Spezielle Anwendungen wie die Update-Programmierung von Kraftfahrzeugen werden mit 802.11p allerdings nicht adressiert. Diese Aufgaben fallen in den Bereich der klassischen W-LAN-Vernetzung. Da derzeit davon ausgegangen wird, dass die Funkmodule für 802.11p ebenfalls die anderen 802.11-Standards unterstützen werden, würden sich entsprechend ausgerüstete Fahrzeuge für Update-Programmierung und Diagnose über W-LAN eignen.

Im Gegensatz zu GSM und UMTS besitzt W-LAN eine geringere Reichweite, die im Freien zwischen 100 m und 300 m und in geschlossenen Räumen bis zu 40 m liegen kann. Mittels Richtantennen lassen sich allerdings auch Entfernungen bis zu einigen Kilometern überbrücken. Jeder W-LAN-Adapter bildet eine Funkzelle für sich. Überschneiden sich zwei Funkzellen, können die beiden gleichwertigen W-LAN-Adapter direkt miteinander kommunizieren. Diese Art der Vernetzung wird als ad-hoc-Vernetzung bezeichnet und ist für W-LAN eher unüblich. Gebräuchlicher ist der Infrastruktur-Modus, bei dem eine Basisstation (Access Point – AP) zum Einsatz kommt. Diese koordiniert die einzelnen Netzknoten und ist meist gleichzeitig die Schnittstelle zu anderen drahtlosen oder drahtgebundenen Netzen.

Die hohen Datenübertragungsraten machen W-LAN im Kraftfahrzeugbereich insbesondere für die Update-Programmierung interessant, bei der hohe Datenraten benötigt werden. Durch die Reichweite, die W-LAN erreicht und die mittels Basisstationen vergrößert werden kann, ist beispielsweise eine vollständige Funkabdeckung von Werkstätten oder Kundendiensteinrichtungen möglich. Darüber hinaus lässt sich W-LAN leicht in IT-Strukturen integrieren.

Gegen den Einsatz von W-LAN im Automobilbereich in sicherheitsrelevanten Bereichen wie der Update-Programmierung spricht die beschränkte Sicherheit. W-LAN wurde auf hohe Datenübertragung ausgelegt, das Thema Sicherheit spielte bei der Entwicklung des Standards eine eher untergeordnete Rolle. Der im Standard verankerte Sicherheitsmechanismus Wired Equvialent Privacy (WEP) wurde sehr schnell nach seiner Veröffentlichung kompromittiert und gilt als unsicher. Daher sind technische Ergänzungen wie beispielsweise Wi-Fi Protected Access (WPA) entwickelt wurden, die das Sicherheitsproblem mehr oder weniger gut verkleinern. Eine andere Alternative ist die Verlagerung der Sicherheit in höhere Protokollschichten wie beim Einsatz von IP-Sec, die dann aber mehr Overhead und mehr Rechenleistung fordert. Offizieller Nachfolger von WEP ist der Standard 802.11i, der derzeit bei nichttrivialen Passwörtern als sicher gilt. Nachteilig an diesen neuen Verfahren sind seine hohe Komplexität und Rechenaufwand, die eine große Rechenleistung voraussetzen und dies im Widerspruch zur geforderten geringen Stromaufnahme steht.

Es ist davon auszugehen, dass die Vorteile von W-LAN mit der hohen Datenübertragungsrate, dem starken Verbreitungsgrad und dem zukünftigen Fokus auf automobile Anwendungen wie Car-to-Car Communication zu einer zunehmenden Bedeutung in der Kraftfahrzeug-Vernetzung führen werden. Für einen Einsatz im Bereich der Diagnosekommunikation, wie er im Rahmen dieser Arbeit

betrachtet wird, sprechen dagegen die Sicherheit und die relativ hohen Kosten der benötigten Infrastruktur im Vergleich zu Bluetooth, das im nächsten Abschnitt vorgestellt wird. Innerhalb dieser Arbeit wurde sich für Bluetooth als drahtloses Diagnoseprotokoll entschieden und W-LAN aus diesem Grund nicht weiter betrachtet. Beide Protokolle entstammen der IT-Welt und sind modular nach dem ISO/OSI-7-Schichtenmodell aufgebaut. Dadurch können sie relativ einfach und für Anwendungen transparent gegeneinander ausgetauscht werden.

3.4.5 Bluetooth

Ebenso wie W-LAN entstammt das drahtlose Übertragungsprotokoll Bluetooth aus der IT-Welt. Konzipiert als „Kabel-Ersatz"-Protokoll zur Bürokommunikation (Telefon, Headset, Drucker usw.) für kurze Distanzen wurden die Prioritäten bei der Entwicklung auf kostengünstige Hardware, geringen Stromverbrauch und genaue Standards für eine sehr gute Interoperabilität zwischen verschiedenen Herstellern gelegt. Im Gegensatz zu W-LAN stand die Sicherheit im Vordergrund und ist grundlegender Bestandteil der Bluetooth-Spezifikation. Diese wurde von der IEEE für Wireless Personal Area Network (WPAN) als 802.15.1 adaptiert. Verantwortlich für die Entwicklung von Bluetooth ist die Bluetooth Special Interest Group (SIG). Um die Belange der Automobilhersteller und –zulieferer kümmern sich innerhalb der SIG die Car-Working-Group (CWG) und Automotive-Expert-Group (AEG).

Bluetooth sendet ebenso wie W-LAN im 2,4 GHz ISM-Band und erreicht eine maximale Bruttodatenübertragungsrate von 1 MBit/s, von der im asynchronen Modus netto 780 kBit/s übrig bleiben. Mit der aktuellen Spezifikation 2.0 wurde Bluetooth um einen Enhanced Data Rate-Modus (EDR) erweitert, der die maximale Datenrate verdreifacht. Damit können netto bis zu 2,2 MBit/s übertragen werden. Bluetooth sieht drei Sendeleistungsklassen mit 1 mW, 2,5 mW und 100 mW vor, die Reichweiten von 10 m bis 100 m zulassen. Bei einem parallelen Einsatz von Bluetooth und W-LAN können durch die Benutzung des gleichen Frequenzbereichs Kollisionen auftreten, die bei beiden Protokollen zu starken Einbrüchen der Datenübertragungsrate führen. Daher verfügt Bluetooth über Mechanismen zur Erkennung und Vermeidung derartiger Kollisionen. Das Bluetooth-Protokoll unterstützt eine automatische Konfiguration von ad-hoc-Netzwerken, so dass zwei oder mehrere Geräte ohne vorherige Kenntnis voneinander miteinander kommunizieren können. Solch ein Netzwerk wird Piconet genannt und besteht aus einem Master und bis zu sieben Slaves. Der Master eines solchen Piconetzes kann wiederum Slave in einem weiteren Piconet sein. Die Gesamtheit miteinander kommunizierender Piconetze wird als Scatternetz bezeichnet.

Die Interoperabilität zwischen verschiedenen Bluetooth-Geräten wird über Profile gewährleistet, die für bestimmte Anwendungsbereiche festgelegt sind. Wenn eine Bluetooth-Verbindung aufgebaut wird, tauschen die Systeme ihre Profile aus und legen damit fest, welche Dienste sie für den jeweiligen anderen Partner zur Verfügung stellen können und welche Daten oder Befehle sie dazu benötigen. Einer der Schwerpunkte bei der Entwicklung des Bluetooth-Protokolls liegt in der Sicherheit bei der Datenübertragung. Bluetooth bietet einige Sicherheits-mechanismen an, deren Verwendung optional ist. Diese basieren auf dem Konzept von Personal Trusted Devices und beziehen sich grundsätzlich auf gesicherte

Verbindungen zwischen zwei Bluetooth-Geräten. Über die Verbindungsebene hinausgehende Sicherheitskonzepte, die etwa den Benutzer autorisieren, liegen außerhalb des Bereichs, der durch die Bluetooth-Spezifikationen abgedeckt wird. Gesicherte Verbindungen zwischen zwei Bluetooth-Geräten verlangen nach einer identischen PIN (Bluetooth Passkey), die vom Gerät vorgeschrieben oder vom Benutzer vergeben wird. Dann wird in beiden Geräten unter Verwendung von den weltweit eindeutigen Bluetooth-Geräteadressen und Zufallszahlen ein 128 Bit langer Verbindungsschlüssel generiert. Dieser Verbindungsschlüssel kann für jede zukünftige Verbindung zur automatischen Authentisierung mittels Challenge-Response-Verfahren verwendet und außerdem für die Chiffrierung des gesamten Funkverkehrs genutzt werden. Das Sicherheitskonzept von Bluetooth gilt zum heutigen Stand als ausreichend sicher. Die einzigen Sicherheitslücken, die in Zusammenhang mit Bluetooth, speziell bei Handys, bekannt wurden, hatten ihre Ursache in einer fehlerhaften Implementierung des Bluetooth-Protokollstacks.

Die im Vergleich zu W-LAN geringe Datenübertragungsrate wird mittelfristig nur ein untergeordnetes Kriterium sein, denn die SIG wird die neue Bluetooth-Spezifikation „Lisbon“ im Laufe des Jahres 2006 veröffentlichen. Diese sieht in einem ersten Schritt drei Betriebsarten von 4, 8 und 12 MBit/s vor, die unter optimalen Betriebsbedingungen bis 10 MBit/s Nettodurchsatzraten liefern. Langfristig ist eine Bluetooth-Version namens „Seattle“ geplant, die ein Ultra Wideband-Übertragungsverfahren (UWB) nutzen wird. Mit diesem Verfahren werden Übertragungsraten von mehreren hundert MBit/s möglich sein. Ein erster Prototyp, der präsentiert wurde, ist bereits für 110 MBit/s Brutto-Datenübertragungsrate ausgelegt [CT2605].

Im Automobil wird Bluetooth heute schon eingesetzt. Es dient dort meist als drahtlose Kurzstreckenverbindung für Consumer-Geräte aus dem Multimedia-Bereich wie PDA, Freisprecheinrichtung (Headset), Notebook, CD-Player usw. Darüber hinaus gibt es Überlegungen, Bluetooth in Keyless-Entry-Systemen (RKE), Fernsteuerungen für z. B. die Sitzverstellung und Klimaregelung oder für Diagnosezwecke einzusetzen. Für einen drahtlosen Diagnosezugang zum Fahrzeug-Kommunikationsnetzwerk steht Bluetooth dabei in direkter Konkurrenz zu W-LAN. Die Diagnose stellt hohe Ansprüche an Bandbreite und Sicherheit. Bluetooth besitzt gegenüber W-LAN Vorteile bei der Sicherheit. Sie ist direkter Bestandteil der Protokollspezifikation, während bei W-LAN erheblich mehr Aufwand betrieben werden muss. Die Beschränkung auf eine geringe und energiesparende Reichweite erhöht die Sicherheit bei Bluetooth zusätzlich. Der derzeitige Nachteil der im Vergleich zu W-LAN geringeren Bandbreite wird sich in den zukünftigen Bluetooth-Versionen relativieren. Aus diesem Grund wurde der drahtlose Diagnosezugang im Rahmen dieser Arbeit mit Bluetooth realisiert. Im Kapitel 5.5.1 wird detaillierter auf den Aufbau des Bluetooth-Protokolls eingegangen.

Die Realisierung mit Bluetooth stellt dabei nur eine Möglichkeit dar und schließt den Einsatz von W-LAN nicht aus. Der modulare Aufbau beider Protokolle nach ISO/OSI-7-Schichtenmodell erlaubt ein einfaches gegenseitiges Austauschen beider Protokolle. Aber auch ein gemeinsamer Einsatz beider Protokolle ist denkbar, bei dem W-LAN die drahtlose Anbindung an eine IT-Struktur bereitstellt. Der drahtlose Diagnosezugang im Fahrzeug verfügt dagegen über eine Bluetooth-Schnittstelle mit den Bluetooth-eigenen guten Sicherheitseigenschaften. Die Kopplung zwischen IT-

Netzwerk, beispielsweise einer Produktionshalle oder des Kundendienstes, und dem Fahrzeug kann über ein Bluetooth – W-LAN-Gateway geschehen. Durch die Kollisionsvermeidung mit W-LAN, für die Bluetooth spezielle Mechanismen besitzt, können beide Protokolle nebeneinander koexistieren und die jeweiligen Vorteile ausgenutzt werden.

3.5 Datenbus-Architekturen

Der im Kraftfahrzeug stark verbreitete CAN-Bus stößt durch die heutigen und zukünftigen Anforderungen an Bandbreite, Determinismus, Kosten usw. immer mehr an seine Grenzen. Die Einführung von weiteren CAN-Bussen in bestimmten Bereichen des Fahrzeugs kann diese Problematik nicht dauerhaft lösen. Einige der Anforderungen wie hohe Sicherheit und niedrige Kosten stehen im direkten Widerspruch zueinander und können von einem „Universal-Datenbussystem“ allein nicht erfüllt werden. Vielmehr wird der CAN-Bus durch die im vorangegangenen Abschnitt vorgestellten Datenbussysteme ergänzt werden. Zusammen decken heutige Datenbusse in ihrer Datenbandbreite nahezu einen Bereich von sechs Dekaden ab, wie Abbildung 24 zeigt. Die verfügbare Bandbreite und Sicherheit des jeweiligen Datenbussystems stehen dabei im direkten Verhältnis zu seinen Kosten.

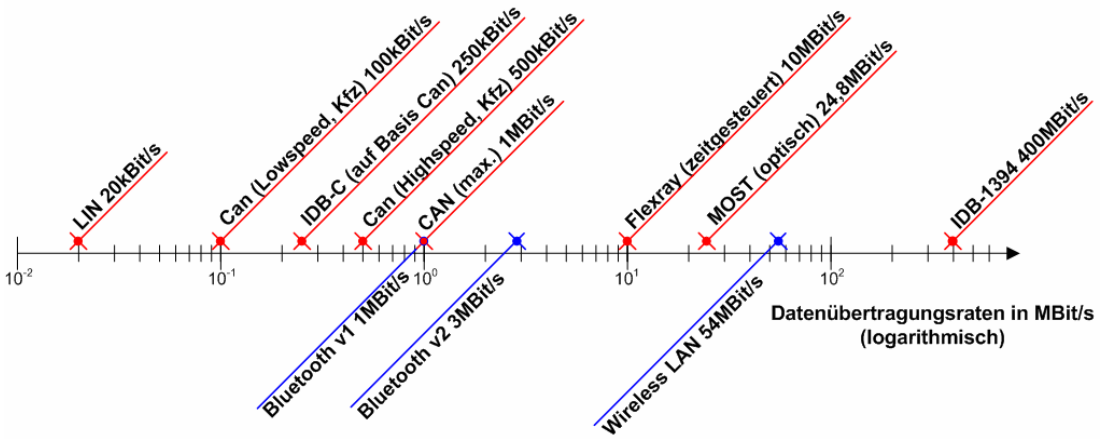

Abbildung 24: Datenübertragungsraten ausgewählter Kfz-Datenbusse

In heutigen Kraftfahrzeugen der Oberklasse können sich auf Grund der erhöhten Komfort- und Sicherheitsbedürfnisse über 70 miteinander vernetzte Steuergeräte befinden. Der dadurch aufkommende starke Kommunikationsbedarf wird durch zukünftige Funktionen weiter zunehmen. Dieses hohe Datenaufkommen im Kraftfahrzeug sowie Kosten- und Sicherheitsüberlegungen machen es unmöglich, die große Anzahl von komplexen Steuergeräten über eine einzige Busleitung zu vernetzen. Durch die Aufteilung der Steuergeräte nach ihren Funktionen mit den jeweiligen Datenübertragungs- und Datenintegritätsbedürfnissen in Domänen ist eine spezifische Vernetzung mit dem jeweiligen optimalen Bussystem möglich. Dabei stellt sich die Frage nach einer geeigneten Datenbusarchitektur, d.h. wie man diese Steuergeräte, Domänen und heterogenen Bussysteme optimiert nach Kosten, Leitungslängen und funktionalen Anforderungen miteinander vernetzt. Für die Auslegung einer solchen Architektur existieren mit einer zentralen und einer dezentralen Variante zwei grundsätzliche Philosophien, die in den nächsten

Abschnitten vorgestellt werden. Zuvor folgt eine kurze Erläuterung über die verschiedenen Möglichkeiten der Kopplung zweier unterschiedlicher Bussysteme.

3.5.1 Kopplung verschiedener Datenbussysteme

Für die Verbindung zweier Datenbussysteme existieren verschiedene Möglichkeiten. Angelehnt an das ISO/OSI-7-Schichtenmodell kann diese Verbindung auf jeder der sieben Ebenen erfolgen. Dabei gilt allgemein, dass je höher die Ebene der Schicht ist, auf der die zwei Datenbusse vernetzt werden, umso mehr Freiheiten und Einfluss besitzt man bezüglich der auszutauschenden Daten. Allerdings steigen gleichzeitig die Komplexität der Umsetzung und damit die benötigte Rechenleistung stark an.

3.5.1.1 Kopplung auf Schicht 1 – Repeater, Hub

Der Repeater wird in der Kommunikationstechnik verwendet, um zwei Segmente eines Datenbussystems physisch miteinander zu einem Netzwerk zu verbinden oder um auf langen Übertragungsleitungen das Signal wieder zu regenerieren. Der Repeater arbeitet auf Ebene 1 des ISO/OSI-7-Schichtenmodells, dem Physical Layer. Daher ist er für andere Netzelemente im Netzwerk, die höhere Schichten bearbeiten, nicht sichtbar. Mit Hilfe des Repeaters lassen sich keine unterschiedlichen Datenbusse miteinander koppeln.

Ähnlich wie der Repeater arbeitet auch der Hub und verbindet Netzwerk-Segmente oder auch einzelne Hubs miteinander. Mit dem Hub lassen sich ebenfalls nur Datenbusse gleichen Typs vernetzen. Der Hub arbeitet, genauso wie ein Repeater, auf Ebene 1 des ISO/OSI-7-Schichtenmodells und wird deswegen auch Multiport-Repeater oder Repeating-Hub genannt. Das Signal eines Netzwerkteilnehmers wird von einem Hub an alle anderen Netzwerkteilnehmer weitergeleitet. Bei Einsatz eines Hubs im Netzwerk wird durch die Verkabelung meist eine Stern-Topologie realisiert, der logische Aufbau eines Hubs entspricht aber einer Bus-Topologie. Dies bedeutet, dass durch einen Hub die maximal zur Verfügung stehende Bandbreite eines Netzwerks nicht gesteigert wird, da sich alle Netzwerkteilnehmer in derselben Kollisionsdomäne befinden. Der Vorteil eines Hubs liegt in der erhöhten Ausfallsicherheit, da eine Störung der Verbindung zu einem Netzwerkteilnehmer nicht das gesamte Netz lahm legt, sondern nur der jeweilige Teilnehmer nicht mehr erreichbar ist.

3.5.1.2 Kopplung auf Schicht 2 – Bridge, Switch

Eine Bridge verbindet in der Kommunikationstechnik zwei Segmente auf der Ebene der Schicht 2 (Sicherungsschicht) des ISO/OSI-7-Schichtenmodells. Eine Bridge kann auf der Unterschicht des Buszugriffs (MAC – engl. Media Access Control) oder der Unterschicht der Rahmenbildung (LLC – engl. Logical Link Control) arbeiten. Eine Bridge wird hauptsächlich eingesetzt, um ein Netz in verschiedene Kollisionsdomänen aufzuteilen. Somit kann die Last in großen Netzwerken vermindert werden, da jeder Netzstrang nur die Pakete empfängt, deren Empfänger sich auch in diesem Netz befindet. Daten, die für alle Geräte bestimmt sind oder keinem bestimmten Empfänger zuzuordnen sind, werden als Broadcast-Daten an alle angeschlossenen Geräte gesendet.

Eine MAC-Bridge verbindet Netze mit gleichen Zugriffsverfahren. Die LLC-Bridge (auch Remote-Bridge oder Translation Bridge) wird verwendet, um zwei Teilnetze mit verschiedenen Zugriffsverfahren (z.B. CSMA/CD und Token-Passing) zu koppeln. Die Busmedien der beiden Teilnetze können ebenfalls unterschiedlich sein. Innerhalb der LLC-Bridge findet eine Umsetzung (Translation) statt. Bei dieser Umsetzung werden alle Parameter des Quellnetzes (wie MAC-Adresse, Größe und Aufbau des MAC-Frames) an das Zielnetz angepasst.

Ein Switch verbindet mehrere Computer (Steuergeräte) bzw. Netzwerk-Segmente in einem lokalen Netzwerk ähnlich einem Hub. Man spricht bei einem Switch auch von einem intelligenten Hub. Der Switch arbeitet in seiner ursprünglichen Form auf der Schicht 2 (Sicherungsschicht) des ISO/OSI-7-Schichtenmodells. Ein Switch wird wegen der ähnlichen Eigenschaften zur Bridge oft auch als Multi-Port-Bridge bezeichnet. Über den Switch lassen sich ebenfalls unterschiedliche Datenbussysteme miteinander vernetzen. Die einzelnen Ports eines Switches können dabei unabhängig voneinander Daten empfangen und senden. Diese sind über einen internen Hochgeschwindigkeitsbus (Backplane) miteinander verbunden. Datenpuffer sorgen dafür, dass nach Möglichkeit keine Datenpakete verloren gehen. Das eigentliche Switching, also die Entscheidung, an welchem Port ein gerade eingetroffener Frame wieder herausgeschickt wird, kann nach verschiedenen Strategien erfolgen, die sich hinsichtlich Aufwand und Leistungsfähigkeit voneinander unterscheiden.

3.5.1.3 Kopplung auf Schicht 3 – Router

Ein Router ist ein Vermittlungsrechner, der in einem Netzwerk dafür sorgt, dass bei ihm eintreffende Daten eines *Protokolls* zum vorgesehenen Zielnetz bzw. Subnetz weitergeleitet werden (Routing). Router arbeiten auf Schicht 3 (Network-Layer) des ISO/OSI-7-Schichtenmodells. Ein Router besitzt für jedes an ihn angeschlossene Datenbussystem, die unterschiedlich sein können, eine Schnittstelle (Interface). Beim Eintreffen von Daten muss ein Router den richtigen Weg zum Ziel und damit die passende Schnittstelle, über welche die Daten weiterzuleiten sind, bestimmen. Dazu bedient er sich einer lokal vorhandenen Routingtabelle. An einem Router enden sowohl die Kollisions- als auch Broadcastdomäne und werden dadurch entkoppelt. Man unterscheidet außerdem Ein- und Mehrprotokoll-Router. Einprotokoll-Router können nur in homogenen Umgebungen eingesetzt werden, so dass diese für den Einsatz im Fahrzeug mit seinen heterogenen Kommunikationsnetzwerken nicht in Frage kommen. Router arbeiten medienunabhängig aber protokollabhängig – bei einer Bridge ist dies genau umgekehrt. Medienunabhängig bedeutet, dass die Schnittstellen eines Routers Teil unterschiedlicher Netzwerke wie CAN, FlexRay, aber auch W-LAN oder Bluetooth sein können. Die Protokollabhängigkeit eines Routers ergibt sich daraus, dass ein Router nur ihm bekannte Protokolle der Schicht 3 des ISO/OSI-7-Schichtenmodells weiterleiten kann. Deutlich zu trennen sind die Begriffe Router und Gateway. Während ein Router ein bestimmtes Protokoll weiterleitet, erfolgt bei einem Gateway die Umsetzung eines Protokolls in ein anderes.

3.5.1.4 Kopplung auf Schicht 7 – Gateway

Ein Gateway erlaubt es Netzwerken miteinander zu kommunizieren, die auf völlig unterschiedlichen Protokollen basieren. Zu diesem Zweck nimmt ein Gateway eine Protokollumsetzung vor. Dem Gateway ist dabei alles erlaubt, was zur Konvertierung der Daten notwendig ist, auch das Weglassen von Informationen, wenn diese im Zielnetz nicht transportiert werden können. Im Detail werden sämtliche Protokollinformationen, die an ein Datenpaket angehängt wurden (z.B. CAN-ID), entfernt und durch andere (z.B. TCP/IP-Informationen zur Ethernet-übertragung) ersetzt. Gateways benötigen im Gegensatz zu Routern, die nur auf der dritten Schicht des ISO/OSI-7-Schichtenmodells arbeiten, Implementierungen aller relevanter Schichten (1-7) des ISO/OSI-7-Schichtenmodells. Für die Anforderungen im Kraftfahrzeug stellt ein Gateway meist die sinnvollste Art der Vernetzung von Datenbussen dar.

Gateways bilden die aufwendigste und eine sehr komplexe Kopplung verschiedener Datenbusse. Dafür bieten sie eine vollständige Unabhängigkeit der jeweils eingesetzten Topologien und Physical Layer. Zusätzlich erlauben Gateways, da sie auf der Applikationsschicht arbeiten, während der Umsetzung einen Zugriff auf die tatsächlichen realen Dateninhalte und nicht nur auf einzelne Protokollsegmente. Dies ermöglicht eine „intelligente“ Umsetzung und Entlastung der Datenbusse durch Filterung der Daten. Allerdings benötigt solch eine Filterung und Verarbeitung der umzusetzenden Daten zusätzliche Rechenleistung.

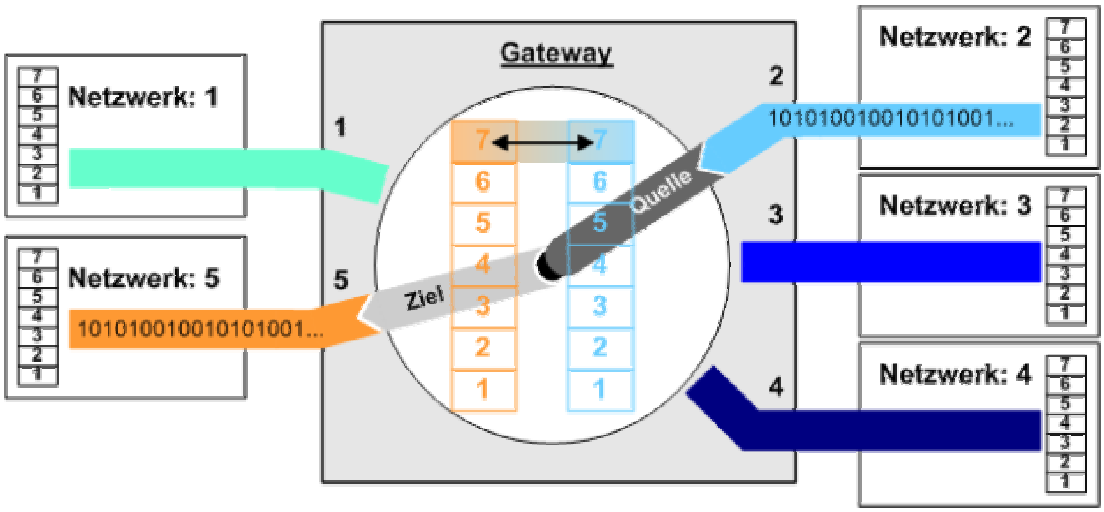

Abbildung 25: Daten- und Protokollumsetzung über ein Gateway

3.5.2 Gateway-Architektur

Eine zentrale Architektur zur Vernetzung der verschiedenen Bussysteme im Kraftfahrzeug wird als Gateway-Architektur bezeichnet. Sie findet sich in nahezu allen aktuellen Fahrzeugen wieder. Diese Architekturform ist historisch gewachsen, indem durch das erhöhte Datenaufkommen zusätzliche Datenbusse, meist CAN, verbaut und diese über ein zentrales Steuergerät, dem Gateway-Steuergerät, verbunden wurden [Vaßen05]. Typischerweise stellt dieses zentrale Gateway-

Steuergerät ebenfalls den Zugang von außen, die Diagnoseschnittstelle, bereit. Abbildung 26 zeigt schematisch den Aufbau einer solchen Architektur.

Die Vorteile der Gateway-Architektur liegen insbesondere in der historischen Entwicklung dieser Architektur, den damit verbundenen langjährigen Erfahrungen mit ihr und den darauf abgestimmten Entwicklungsprozessen. Die direkte Umsetzung der Daten von einem Bussystem auf ein anderes hält die Datenlaufzeiten in Abhängigkeit von der verfügbaren Rechenleistung des Gateways relativ gering. Da im Gateway-Steuergerät alle Daten der angeschlossenen Datenbusse des Fahrzeugs zusammenlaufen, kann über ein derartiges Gateway ein einfacher Zugriff auf diese Bussysteme von außen ermöglicht werden. Dieser einfache Zugriff ist ein wichtiger Vorteil der Gateway-Architektur. Die derzeit im Fahrzeug eingesetzten Protokolle sind meist speziell für den CAN-Bus optimiert und sehen keinerlei Routing der Daten über verschiedene Busse vor. Dadurch lässt sich die Kommunikationssoftware der einzelnen angeschlossenen Steuergeräte einfach halten, da das erforderliche Routing vollständig vom Gateway-Steuergerät übernommen wird.

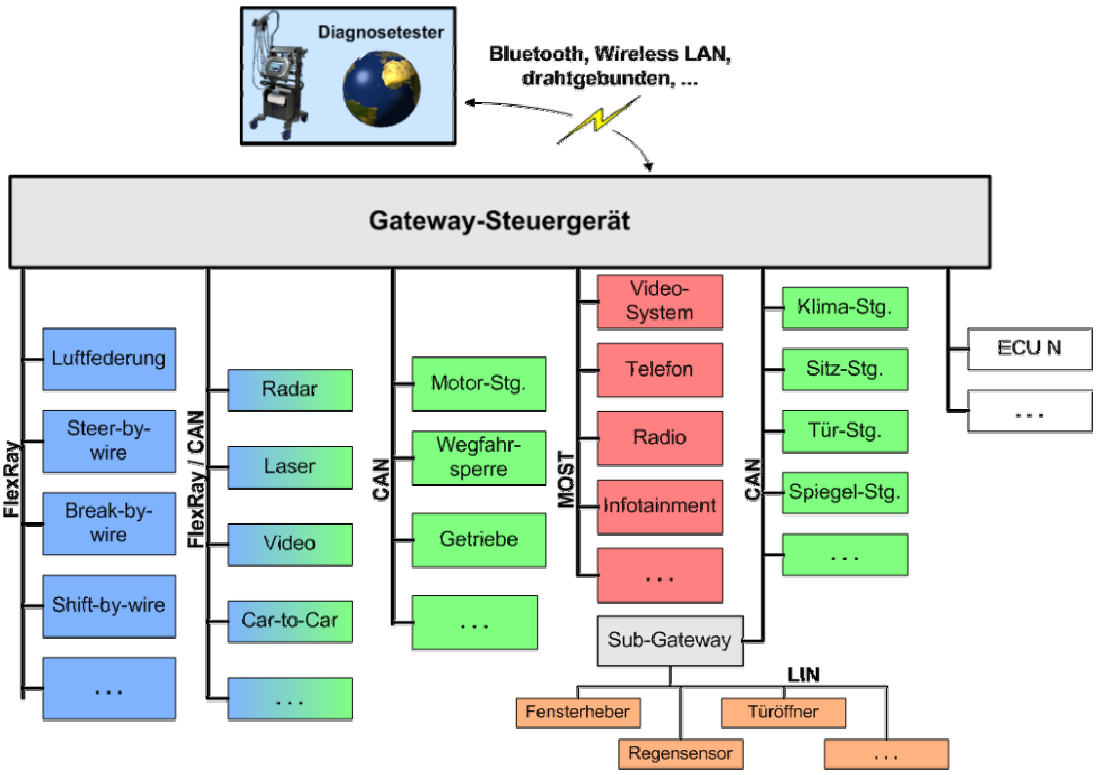

Abbildung 26: Gateway-Architektur im Fahrzeug (Prinzip)

Das Routing im Gateway-Steuergerät wird mittels stark anwendungsspezifischer proprietärer Verbaulisten/Verbindungsmatrizen vorgenommen. Mit der wachsenden Anzahl der Datenbusse in einem Fahrzeug steigen auch die Anforderungen, die an ein solches Gateway gestellt werden. Die einzelnen Protokollumsetzungen von verschiedenen CAN-Bussen, MOST, LIN usw. verlangen eine hohe Rechenleistung. Durch den Einzug von zeitgesteuerten Datenbussen wie FlexRay wird diese Problematik noch deutlich verstärkt werden. Durch die Vielzahl der unterschiedlichen Datenbusse in aktuellen Kraftfahrzeugen wird das Gateway-Steuergerät auch als „Super-Gateway“ bezeichnet.

Die Steuergerätesoftware auf einem Gateway mit ihren verschiedenen unterstützten Protokollstacks (N Protokolle) wird zunehmend komplexer und damit zwangsläufig fehleranfälliger. Ein zentrales Gateway stellt einen Single Point of Failure für das Fahrzeug-Kommunikationssystem dar, d.h. ein Fehler im Gateway wird sich sofort auf das gesamte Datennetz des Fahrzeugs auswirken. Die Entwicklungskosten für ein solches Gateway-Steuergerät sind sehr hoch, da es in höchstem Maße fahrzeugspezifisch ist und dementsprechend viele Varianten entstehen. Diese Kosten und die Komplexität verhindern den Einsatz eines universellen Super-Gateways. Es existieren vielmehr eine große Anzahl baureihenoptimierter, kaum skalierbarer und nicht wieder verwendbarer Gateways.

3.5.3 Backbone-Architektur

Der zentralen Gateway-Architektur steht mit der Backbone-Architektur ein dezentraler Ansatz gegenüber, der schematisch in Abbildung 27 skizziert ist. Der dezentrale Ansatz erlaubt eine Trennung zwischen räumlicher und logischer Verteilung der Funktionen. Er entsteht durch gedankliches Aufbrechen und Auseinanderziehen des zentralen Gateways. Die Protokollumsetzung geschieht dabei in mehreren kleinen Gateways oder in bereits vorhandenen Steuergeräten, deren Software um eine zusätzliche Gateway-Funktionalität erweitert wird. Diese Gateways sind über einen Datenbus mit hoher Bandbreite, dem Backbone, miteinander verbunden.

Dieser dezentrale Ansatz hat sich bislang im Kraftfahrzeug noch nicht durchsetzen können. Ziel dieser Arbeit ist die prototypische Implementierung einer solchen Architektur für ein Kraftfahrzeug-Netzwerk. Es wird damit die prinzipielle Funktionsfähigkeit eines solchen Ansatzes gezeigt und Anforderungen sowie notwendige Voraussetzungen herausgearbeitet. Der Nachweis der verbesserten Leistungsfähigkeit hinsichtlich zukünftiger Ansprüche an Durchsatz, Flexibilität, Kosten und beherrschbarer Komplexität gegenüber der Gateway-Architektur wird durch die Update-Programmierung von Steuergeräten über das Kommunikationssystem geführt.

Im Bereich der IT-Kommunikation ist die Backbone-Architektur bereits etabliert. Dort werden die Netzwerksegmente (Domänen) einzeln für sich vernetzt, beispielsweise die Ethernetvernetzung von Etagen eines Bürohauses. Die Art der jeweiligen Etagenvernetzung spielt dabei für das Gesamtnetzwerk keine Rolle. Über einen Backbone hoher Bandbreite, der z.B. anstelle des Ethernet-Protokolls das ATM-Protokoll nutzt, werden die einzelnen Segmente, im Beispiel die Etagen, miteinander verbunden. Die Vorteile sind die einfachere und strukturierbare Verkabelung, die höhere Stör- und Ausfallsicherheit bei Fehlern in den Netzsegmenten sowie die hohe Flexibilität bei Änderungen. Bezogen auf das Beispiel der Bürohausvernetzung bedeutet der letzte Punkt bei einem Umzug einer Organisationseinheit auf eine andere Etage (Domäne) im einfachsten Fall das Umstecken eines Steckers, um dieses Segment in eine neue Domäne zu befördern.

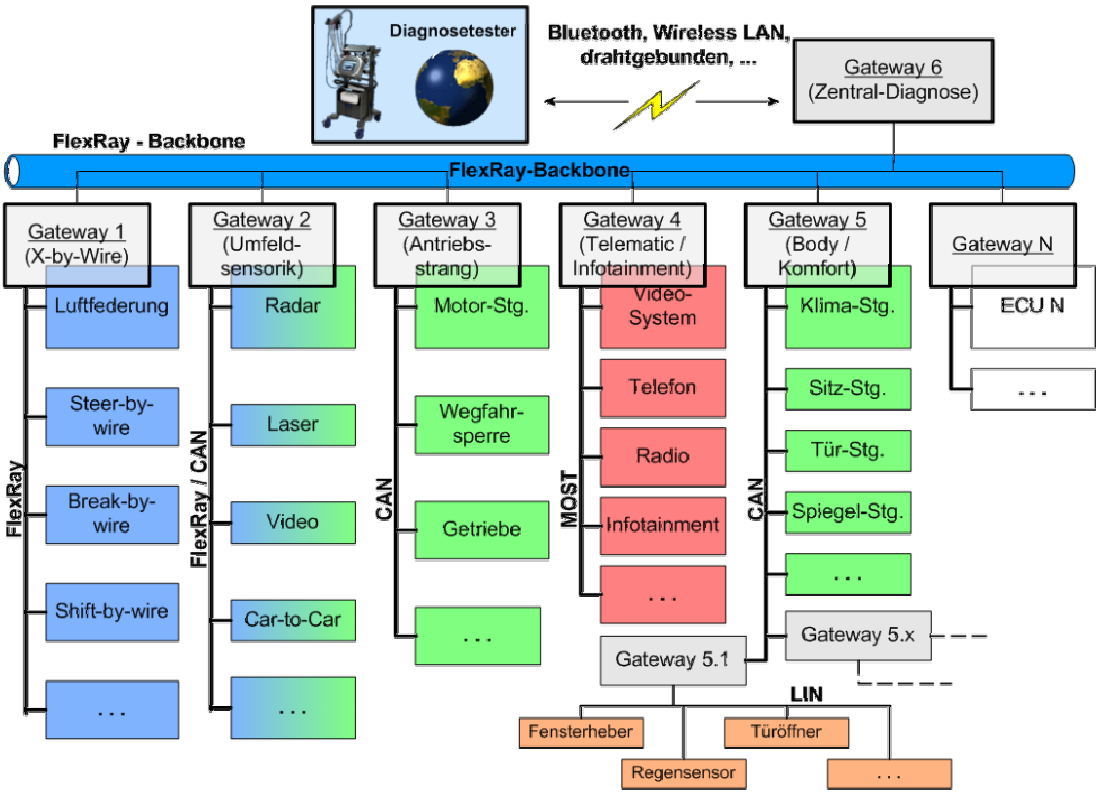

Abbildung 27: Backbone-Architektur im Fahrzeug mit FlexRay (Prinzip)

Ähnliche Vorteile, wie sie die Backbone-Architektur für die IT-Vernetzung bringt, ergeben sich auch beim Einsatz im Kraftfahrzeug. Die Backbone-Architektur entkoppelt die einzelnen Fahrzeug-Domänen voneinander. Diese können dann räumlich begrenzt bleiben und benötigen keine direkte Verkabelung mit dem zentralen Gateway mehr, was Fehlerquellen, Verkabelungsaufwand und Kosten reduziert. Auftretende Fehler innerhalb der Domänen oder in den Gateways stören die übrige Kommunikation des Fahrzeugdatennetzes nicht. Das Fahrzeug bleibt in Grenzen funktionsfähig. Bei Änderungen in der Ausstattung des Fahrzeugs sind Umverteilungen, Entfall oder zusätzlicher Einbau einzelner Steuergeräte mit geringem Aufwand möglich, da sie nicht mehr das Gesamtnetzwerk, sondern nur die einzelne Domäne betreffen.

Neben diesen allgemeinen Vorzügen der Backbone-Architektur ergeben sich weitere fahrzeugspezifische Vorteile. Die Komplexität, die ein zentrales Gatewaysteuergerät nach sich zieht, wird durch kleinere und einfachere Gateways signifikant verringert. Insbesondere die Software gewinnt an Stabilität durch die Umsetzung zweier Datenprotokolle anstelle von N Datenprotokollen einer zentralen Gateway-Steuergerätesoftware. Die Entwicklungskosten sinken durch die Modularität solcher kleiner Gateways, die zum überwiegenden Teil nur noch aus Software bestehen. Die Modularität schränkt gleichzeitig die Variantenvielfalt der Gateways auf ein überschaubares Maß ein, da diese nur noch für die einzelnen Datenprotokollvarianten benötigt werden. Damit sind die Gateways und ihre Software unabhängig vom Fahrzeugmodell, in dem sie eingesetzt werden. Verglichen mit dem Super-Gateway bei der zentralen Gateway-Architektur erlaubt die Modularität der Backbone-Architektur zusätzlich eine bessere Skalierbarkeit. Wird erst gegen Ende des

Entwicklungszyklus eines Fahrzeugmodells deutlich, dass einer der eingesetzten Datenbusse überlastet ist, muss dieser durch einen Datenbus höherer Bandbreite ausgetauscht werden. Bei einem Super-Gateway würde diese Lösung ein komplettes Redesign des Gateways und der an den Datenbus angeschlossenen Steuergeräte erforderlich machen, während bei Einsatz der Backbone-Architektur sich das eingesetzte Gateway austauschen ließe und die Änderungen sich allein auf die Domäne beschränkten. Eine alternative Lösung zur Behebung der Überlast wäre die Umverteilung der Steuergeräte auf andere Datenbusse zur Entlastung. Auch hierbei bietet die Backbone-Architektur durch ihre Flexibilität die größeren Vorteile gegenüber der Gateway-Architektur.

Einer der Nachteile der Backbone-Architektur liegt in der benötigten Anzahl an zusätzlichen Gateway-Steuergeräten. Da die Gateways, die bei der Backbone-Architektur zum Einsatz kommen, zum überwiegenden Teil aus überschaubaren Softwaremodulen bestehen, lässt sich dieser Nachteil durch Funktionsintegration in bereits vorhandene Steuergeräte entschärfen. Diese übernehmen dann zusätzlich zu ihren eigentlichen Aufgaben die Gateway-Funktionalität für ihre Domäne. Zwei größere Nachteile der Backbone-Architektur gegenüber des Super-Gateways existieren dennoch. Bei der domänenübergreifenden Kommunikation treten im Vergleich zur Gateway-Architektur geringfügig längere Latenzzeiten auf, da die Daten über mindestens zwei Gateways anstelle eines zentralen Gateways umgesetzt werden müssen. Diese Latenzen sind allerdings im Vorfeld in Grenzen bekannt und können somit bei der Auslegung des Kommunikationssystems berücksichtigt werden.

Die wesentliche Einschränkung der Backbone-Architektur liegt im Entfall der Routing-„Intelligenz“ des zentralen Super-Gateways. Dieses hat über Verbindungsmatrizen und Verbaulisten die Anordnungen der im Fahrzeug befindlichen Steuergeräte und Datenbusse gespeichert. Damit ist das Gateway in der Lage, Daten gezielt in andere Netze zu vermitteln. Die Backbone-Architektur verfügt dagegen nur über vergleichsweise einfache Gateways, deren einzige Aufgabe es ist, Daten zwischen zwei bestimmten Datenbussen zu vermitteln. In einem einfachen Beispiel würde das die Vermittlung zwischen Domänen-Datenbus und dem Backbone bedeuten. Würden dabei alle Daten, die über den Backbone transportiert werden, von jedem Gateway auf seinen jeweiligen Domänen-Datenbus umgesetzt, wäre ein Großteil der Vorteile einer Backbone-Architektur zunichte.

Die Lösung dieses Problems stellt die Verlagerung der statischen, da fest programmierten, und damit unflexiblen Intelligenz eines Super-Gateways in eine untere Schicht des ISO/OSI-7-Schichtmodells dar. Als Vorbild kann dazu das Internet mit dem IP-Protokoll dienen, das auf der Schicht 3 die Vermittlung der Datenpakete vornimmt. Da die Datenraten im Kraftfahrzeug etwas geringer und die Anforderungen etwas anders gelagert sind, muss ein automobiles Routing-Protokoll entsprechend angepasst werden. Dem Nachteil des bei der Nutzung eines Routing-Protokolls etwas höheren Overheads stehen die Vorteile einer dynamischen Lösung mit höherer Flexibilität und der Integration in bereits bestehende Transportprotokolle gegenüber. Eine mögliche Umsetzung wird in Abschnitt 6.3 beschrieben und diskutiert.

Eine der wichtigsten Kriterien bei Einführung der Backbone-Architektur ist die Frage nach einem geeigneten Backbone-Datenbus. Seine Anforderungen sind in

erster Linie hohe Bandbreite und Flexibilität. In Voraussicht auf zukünftige Anwendungen wie X-by-Wire spielen aber auch Sicherheit und harte Echtzeitfähigkeit eine Rolle. Wird der CAN mit maximal 1 MBit/s als Bezug genommen und die Datenrate als Hauptkriterium betrachtet, kommen als Backbone MOST, IDB 1394 und FlexRay in Frage. MOST scheidet dabei durch seine aufwendige Ring-Topologie und seinen kostenintensiven optischen Physical Layer aus. Für IDB 1394 sprechen die sehr hohe Datenübertragungsrate, die genug Reserven für ein Echtzeitprotokoll bietet [Beckmann02], und die hohe Verbreitung in der Consumerelektronik. Die Nachteile bei IDB 1394 sind das derzeit nur prototypisch Vorhandensein im Fahrzeug und damit verbundene unzureichende Erfahrungen bezüglich Sicherheitsreserven und Störanfälligkeit im Fahrzeug. In dem Zusammenhang wird auch Ethernet als mögliche Backbone-Alternative diskutiert. Für Ethernet gelten die gleichen Voraussetzungen wie für IDB 1394. Derzeit existieren allerdings noch keine ernsthaften Bestrebungen seitens der Automobilindustrie, Ethernet in das Fahrzeug zu bringen.

FlexRay bietet gegenüber IDB 1394 oder Ethernet eine relativ geringe Bandbreite von 10 MBit/s, stellt allerdings in der nächsten Generation eine Verzehnfachung dieser Datenübertragungsrate in Aussicht. Darüber hinaus besitzt FlexRay den großen Vorteil, von der Automobilindustrie für den Einsatz im Kraftfahrzeug entwickelt zu sein und auch eingesetzt zu werden. Ausgelegt als zeitgesteuerter Datenbus für sicherheitskritische Anwendungen verfügt FlexRay daneben über eine hohe Flexibilität, die ihn für den Einsatz als Backbone geeignet erscheinen lassen. Aus diesem Grund wird in dieser Arbeit der Backbone mit FlexRay realisiert.

4 Diagnose und Update von Steuergerätesoftware (Flash)

Wie in den vorangegangen Abschnitten schon angedeutet, setzt die Automobilindustrie aus den verschiedensten Gründen verstärkt auf Steuergeräte, die auch nachträglich wiederholt reprogrammiert werden können. Die notwendige Voraussetzung für eine mehrfache Programmierung ist der Einsatz von Flash-Speichern anstelle des Masken-ROM. Aber gerade der Austausch der auf den ersten Blick preiswerten Masken-ROM-Bausteine durch teurere Flash-Bausteine ist bei den Automobilherstellern auf Grund von Kosten- und Sicherheitsaspekten umstritten. Im Rahmen dieser Arbeit wurde daher eine Untersuchung [Torney04] durchgeführt, die diese Problematik beleuchtet und deren Ergebnisse zusammengefasst hier wieder gegeben werden. Die Betrachtung erfolgt nicht auf der Kostenebene, dem Preis, sondern berücksichtigt mit dem Preis-Leistungsverhältnis auch die technischen Möglichkeiten des Flash-Speichers gegenüber des Masken-ROMs.

Des Weiteren werden bei der Update-Programmierung im Kraftfahrzeug spezielle Prozesse benötigt, die eine Programmierung der Steuergeräte über den angeschlossenen Datenbus erlauben. Gleichzeitig müssen diese Prozesse sicherstellen, dass nur autorisierte Programmierungen durchgeführt werden können und diese die Funktion des Steuergeräts nicht beeinträchtigen. Die Verfahren, die sich auf dem Gebiet etabliert haben, werden in dem Kapitel kurz vorgestellt und ihre Beschränkungen hinsichtlich einer heterogenen Kommunikationsumgebung aufgezeigt.

4.1 Anforderungen an automobile Flash-Speicher

Die Anforderungen, die an automobile Flash-Speicher gestellt werden, setzen sich aus allgemeinen Forderungen für Speicherbausteine und speziellen Kriterien, wie sie im Automobilbereich gefordert werden, zusammen. Für eine entsprechende Systematik wurden die identifizierten Anforderungen in die Punkte der physikalischen Anforderungen, Zuverlässigkeit, Verfügbarkeit und Sicherheit klassifiziert, die im Folgenden beschrieben sind. Zusätzlich spielen aber auch betriebswirtschaftliche Kriterien wie Preis, Lieferbedingungen und Ersatzteilhaltung eine Rolle. Diese Anforderungen werden je nach Philosophie der Fahrzeughersteller unterschiedlich gewichtet, was eine pauschale Beurteilung erschwert.

4.1.1 Physikalische Anforderungen

An jede im Kraftfahrzeug verbaute Komponente werden hohe physikalische Anforderungen gestellt, z.B. bezüglich Erschütterungen, Temperaturfestigkeit oder Schutz gegen Feuchtigkeit. Um einen sicheren Betrieb der Steuergeräte gewährleisten zu können, muss der Flash-Speicher für diese rauen Umgebungsbedingungen ausgelegt und angepasst werden. Eine besondere Rolle für die Flash-Speicher spielen bei den physikalischen Anforderungen der Temperaturbereich und die starken Temperaturschwankungen, in dem der Speicher jederzeit funktionieren

muss. Typischerweise liegt der Temperaturbereich für elektronische Bauteile im Kraftfahrzeug zwischen -40°C und +150°C, wobei noch zwischen den Anwendungen unterschieden werden kann [Infineon03]. So sind beispielsweise Steuergeräte im Antriebsstrang wie beim Motor auf Grund der zahlreichen Schnittstellen zu den Sensoren und Aktoren sehr nahe am Aggregat verbaut und damit einen sehr hohen Temperaturbereich ausgesetzt [Schäuffele03]. Typische Temperaturanforderungen sind in Tabelle 4 aufgeführt.

Temperaturbereich	Typische Anwendungsbereiche im Kfz
- 40 °C bis + 85 °C	Bodyelektronik, Komfortbereich
- 40 °C bis + 125 °C	Antrieb, Motor, Getriebe
- 40 °C bis + 150 °C	Direkt am oder im Aggregat verbaute Mikrohybridplatte

Tabelle 4: Typische Temperaturanforderungen im Kfz-Bereich

Neben der Temperaturbeständigkeit ist die elektromagnetische Verträglichkeit (EMV) ein sehr wichtiger Aspekt. Sicherheitsrelevante Anwendungen oder der Rundfunk- und Mobilfunkempfang dürfen durch elektromagnetische Störeinflüsse von außen, die durch den erheblichen Anteil an Elektroniksystemen im Fahrzeug sehr hoch sein können, nicht in ihrer Funktion gestört werden. Aus diesem Grund müssen die EMV-Anforderungen bezüglich Ein- und Abstrahlung sowie entsprechende EMV-Maßnahmen schon bei der Entwicklung des Steuergeräts berücksichtigt werden. Dies bedeutet für die Flash-Speicher, dass er unempfindlich gegenüber Einstrahlungen bis zu einem geforderten Grenzwert sein muss. Darüber hinaus darf er selbst möglichst wenig elektromagnetische Störungen abstrahlen, um andere Bausteine nicht störend zu beeinflussen.

4.1.2 Verfügbarkeit und Zuverlässigkeit

Die Verfügbarkeit ist ein Maß für die Wahrscheinlichkeit, ein System zu einem vorgegebenen Zeitpunkt in einem funktionsfähigen Zustand anzutreffen. Sie ist besonders wichtig für zeitkritische Anwendungen, die Daten in einem vorgegebenen Zeitraum benötigen. Demgegenüber steht als weiteres Maß die Ausfallwahrscheinlichkeit, die meist für Flash-Speicher spezifiziert wird. Sie kann prozentual (ppm, engl. parts per million) oder absolut (fit, engl. failure in time) angegeben werden. Üblicher ist die prozentuale Angabe, die sich bei Flash-Speicher um Werte kleiner als 1 ppm bis 2 ppm bewegt [Kaindl03]. Bei den Werten der Ausfallwahrscheinlichkeit ist zu beachten, dass diese sich nicht direkt auf den Flash-Speicherbaustein, sondern sich meist auf das gesamte System beziehen (z.B. beim integrierten Flash-Speicher ist der Mikrocontroller mit zu berücksichtigen). Die Ausfallwahrscheinlichkeiten der einzelnen Komponenten werden dabei aufsummiert und müssen daher weit unter der geforderten Ausfallwahrscheinlichkeit liegen. Es wird angestrebt, die Ausfallwahrscheinlichkeit weiter zu reduzieren, wobei große Chiphersteller schon heute eine „Null-Fehler-Strategie“ praktizieren und einen großen Teil der Mikrocontroller mit integriertem Flash-Speicher mit einer Ausfallwahrscheinlichkeit von 0 ppm fertigen [Loch04].

Zuverlässigkeit ist definiert als die Erfüllung von bestimmten Eigenschaften unter vorgegebenen Bedingungen für ein gegebenes Zeitintervall [Schäuffele03]. Bezogen auf die Flash-Speicher bedeutet dies, dass gespeicherte Daten auch bei großen Belastungen im Fahrzeug wie hohe Temperaturen oder Spannungsschwankungen über die gesamte Lebensdauer nicht verloren gehen dürfen. Diese Eigenschaft der Datenerhaltung wird bei Flash-Speichern als „Data Retention“ bezeichnet und stellt insbesondere im Fahrzeug ein sehr wichtiges Qualitätsmerkmal dar. Da eine erhöhte Temperatur immer eine Beschleunigung des in diesem Fall unerwünschten Tunneleffektes von Elektronen vom Floating Gate durch die Oxidschicht zur Folge hat, wird die Data Retention immer in Zusammenhang mit einer höchst zulässigen Betriebstemperatur angegeben. So ist bei derzeit erhältlichen Flash-Speichern für den Automobilbereich eine übliche Größe für den garantierten Datenerhalt von 20 Jahren bei 125 °C. Dies bedeutet, dass programmierte Daten für mindestens 20 Jahre bei einer Dauerbetriebstemperatur von 125 °C erhalten bleiben. Mit sinkender Temperatur erhöht sich der garantierte Datenerhalt entsprechend [Münster99]. Bei einem Kraftfahrzeug wird von einer durchschnittlichen Lebensdauer von 10 bis 15 Jahren ausgegangen, was gleichzeitig die Forderung für den garantieren Datenerhalt von Flash-Speichern darstellt.

Neben der Temperatur haben die Anzahl der durchgeführten Schreib- und Löschzyklen einen großen Einfluss auf den Datenerhalt. Die angegebene Eigenschaft der Data Retention bezieht sich daher immer gleichzeitig auf eine maximale Anzahl derartiger Zyklen. Hierbei muss zwischen Flash-Speicher mit und ohne EEPROM-Emulation unterschieden werden. Unter Emulation versteht man die Nachbildung eines Systems durch ein anderes System, wobei in diesem Fall der EEPROM aus Kostengründen durch den Flash-Speicher nachgebildet wird. Bei Flash-Speichern ohne Emulation werden z.B. von BMW mindestens 1000 Schreib- und Löschzyklen für jede Speicherzelle bei einem Datenerhalt von zehn Jahren und einer Ausfallwahrscheinlichkeit von kleiner 2 ppm gefordert [Kaindl03]. Es kann davon ausgegangen werden, dass der Großteil der Steuergeräte nicht häufiger als 20-mal geflasht wird, so dass diese Werte für Flash-Speicher ohne Emulation unkritisch sind.

Bei Flash-Speichern mit EEPROM-Emulation werden dagegen meist eine Million Zyklen gefordert. Die Auslegung erfolgt anhand der maximal zu schreibenden Daten wie beispielsweise des Kilometerstands eines Post-Fahrzeugs. Die geforderten Größenordnungen erreichen herkömmliche Flash-Speicher in Wirklichkeit nicht. Das Ausdauerverhalten kann derzeit nur durch die Bereitstellung mehrerer Blöcke realisiert werden (z.B. 10 Blöcke á 100.000 Zyklen). Wichtig für Flash-Speicher mit EEPROM-Emulation ist die Möglichkeit des gleichzeitigen Schreib- und Lesezugriffs (engl. simultaneous read/write) auf den Speicher. Damit lassen sich Daten aus einem Speicherblock lesen, während ein anderer Speicherblock desselben Speicherbausteins aktualisiert wird. Da im EEPROM Daten mit hohen Änderungszyklen abgelegt sind, muss auch beim Flash-Speicher während der Ausführung von Anwendungen gewährleistet sein, dass dieser beschrieben werden kann.

4.1.3 Programmierung

Eine der wesentlichen Eigenschaften der Flash-Speicher für den Prozess der Update-Programmierung im Automobil ist die maximale Dauer des Programmiervorgangs. Ausgehend von einem idealen Datenzugang unbegrenzter Bandbreite stellt sie den bestimmenden Faktor im Zeitverhalten des Prozesses dar. Mit modernen Flash-Speichern sind zurzeit maximale Programmierdatenraten von bis zu 20 MBit/s möglich, wobei es hier einen erheblichen Unterschied zwischen NAND- und NOR-Speicher gibt[21] [Leitner04]. Bei der Anbindung der Flash-Speicher über Datenbusse sind allerdings die benutzten Protokolle zu berücksichtigen, die einen teils erheblichen Overhead für die reinen Programmierdaten darstellen. Aus diesem Grund sind die möglichen Programmierzeiten aktueller Flash-Speicher momentan noch kein kritisches Kriterium für den Einsatz im Fahrzeug.

Ein anderes Merkmal, was bislang nach Erkenntnissen der durchgeführten Untersuchung noch zuwenig Beachtung findet, ist die benötigte Programmierspannung des Flash-Speichers. Diese muss für eine zuverlässige Speicherung der Daten über die gesamte Dauer der Programmierung möglichst konstant bleiben, was insbesondere im Fahrzeug durch die Spannungsquelle „Batterie“ problematisch sein kann. Fällt die Spannung während der Programmierung kurze Zeit ab, kann es passieren, dass zu wenig Ladungsträger in die Speicherzelle gelangen. Das Gefährliche an dieser Situation ist, dass diese Problematik von außen nicht diagnostizierbar ist. Eine solche Speicherzelle kann eine bestimmte Zeitlang – einige Stunden bis zu einigen Tagen – gültige Daten liefern, dann aber durch Tunneleffekte soviel Ladung verlieren, dass sich ihr Zustand verändert und damit die gespeicherten Daten unbrauchbar macht. Hier wäre eine integrierte Spannungsüberwachung in dem Flashbaustein sinnvoll.

4.1.4 Sicherheit und Schutz

Ein weiterer wichtiger Punkt beim Einsatz von Flash-Speicher stellt die Sicherheit der gespeicherten Daten dar. Diese bezieht sich dabei auf den Schutz vor unbeabsichtigtem Überschreiben und vor nicht autorisiertem Manipulieren und Kopieren der Daten. Der Schutz der Daten gegen Manipulation ist wichtig, um deren Integrität und damit die Funktionsfähigkeit der Anwendung zu gewährleisten. Die Möglichkeit neuer Geschäftsfelder mit zusätzlichen Softwareprodukten, die nachträglich und temporär programmiert und freigeschaltet werden können („Software als Produkt“), verlangt nach einem Schutz, der verhindert, dass diese Daten unerlaubt kopiert und weitergegeben werden.

Um den nicht autorisierten Zugriff auf gespeicherte Daten zu verhindern, werden neben Hardwaremaßnahmen auf den Flash-Speichern vor allem kryptographische Algorithmen eingesetzt. Diese basieren auf digitalen Signaturen („Keywords“), die auf dem Flash-Speicher abgelegt werden und über die eine gültige Autorisierung sichergestellt werden kann. Darüber hinaus lassen sich die kryptographischen Verfahren auch zur Verschlüsselung der Daten auf dem Flash-Speicher nutzen.

[21] Beispielswerte Toshiba: NAND ca. 20 MBit/s; NOR ca. 1 MBit/s [Högl03]

Prinzipiell lassen sich drei verschiedene Stufen des Schutzes (engl. Protection) nutzen [Kaindl03]:

- Read Protection: Der Sektor ist gegen Auslesen geschützt und kann nur nach Autorisierung über eine bestimmte, beispielsweise serielle, Schnittstelle ausgelesen werden. Dies macht z.B. bei einer digitalen Signatur Sinn, die nach ihrer Programmierung vom Flash-Speicher intern genutzt wird.

- Erase/Write Protection: Der Sektor ist gegen Löschen und Programmieren geschützt, um z.B. ein Keyword abzuspeichern, das ohne Autorisierung nicht geändert werden darf.

- One Time Protection (OTP): Der Sektor darf nur einmalig für die Programmierung, beispielsweise einer eindeutigen Seriennummer (engl. unique serial number) verwendet, d.h. die Programmierung ist irreversibel.

4.2 Update-Programmierung entlang der Wertschöpfungskette

Um eine ganzheitliche Aussage über den Nutzen und die Wirtschaftlichkeit der Flash-Speicher und der dadurch möglichen Update-Programmierung treffen zu können, ist zunächst eine Betrachtung des vollständigen Wertschöpfungsprozess eines Kraftfahrzeugs notwendig. Dazu wird in Anlehnung an betriebswirtschaftliche Vorgehensweisen zu Wertkettenanalysen eine Untergliederung des gesamten Unternehmensprozesses in einzelne Teilprozesse vorgenommen [Hentze01]. In Abbildung 28 ist der Produktionslebenszyklus eines Kraftfahrzeugs dargestellt, anhand derer der Einsatz sowie einige der resultierenden Nutzen und Aufwendungen von flashbaren Steuergeräten in den einzelnen Funktionsbereichen gezeigt wird. Zur Vereinfachung wurde eine reduzierte, schematische Darstellung gewählt, die den gesamten Lebenszyklus nicht exakt zeitlich und inhaltlich wiedergibt. Es sind nur die für die Betrachtung der Update-Programmierung direkt relevanten Funktionsbereiche dargestellt. Daher wurden Funktionsbereiche wie beispielsweise Marketing/Vertrieb oder Personalwirtschaft nicht weiter berücksichtigt. Darüber hinaus werden auch die zeitlichen Überschneidungen der verschiedenen Tätigkeiten beispielsweise von Logistik und Produktion oder der durchgehende Beschaffungsprozess vernachlässigt, da es sich um eine qualitative Betrachtung handelt.

Abbildung 28: Schematische Darstellung des Produktlebenszyklus eines Kraftfahrzeugs

4.2.1 Entwicklung

Die immer weiter verkürzten Markteinführungs- und dadurch bedingten verkürzten Entwicklungszeiten führen zu einer immer früher notwendigen Spezifizierung der Systeme und Parallelentwicklung (engl. simultaneous engineering). Die Elektronik-Hardware muss aus produktionstechnischen Gründen zu einem früheren Zeitpunkt in

einem serienreifen Zustand vorliegen als die Softwareentwicklung abgeschlossen sein kann. Während der Entwicklung sowie nach den Integrationstests bis zum Produktionsstart (SOP, engl. Start of Production), und teilweise über diesen hinaus, besteht eine hohe Änderungshäufigkeit der Software, so dass eine Reprogrammierung zum wiederholten Software-Update zwingend notwendig ist. Damit lassen sich Prototypen mittels Update-Programmierung auf dem aktuellen Softwarestand halten. Flashbare Steuergeräte sind die Voraussetzung für eine zeit- und kostensparende Entwicklung.

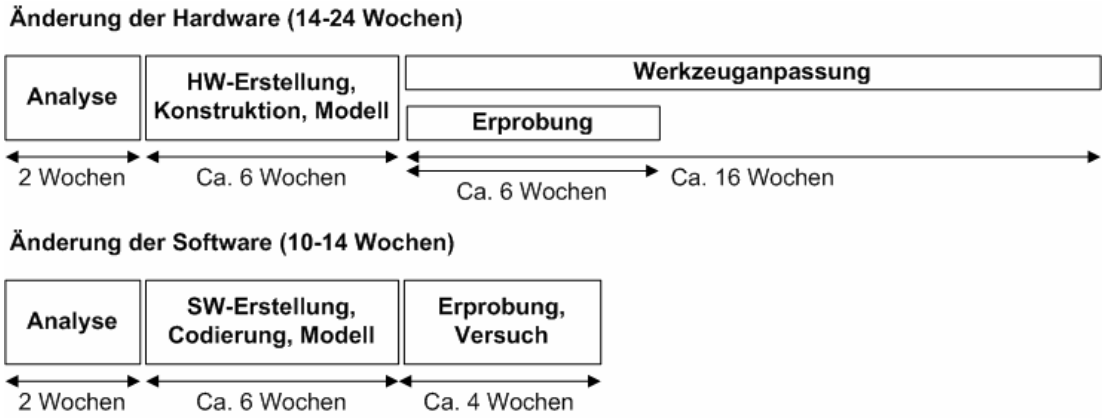

Abbildung 29: Zeitbedarf für Software- und Hardwareänderungen [Faulbacher03]

Einen beispielhaften Vergleich zur Verdeutlichung des zeitlichen Unterschieds zwischen Hardware- und Softwareänderung[22] zeigt Abbildung 29. Der Zeitbedarf für die Analyse und die Erstellung der Änderung zwischen den beiden Alternativen differiert kaum, während ein deutlicher Vorteil bei der Änderungsumsetzung zugunsten der Softwareänderung festzustellen ist. Damit entfallen beispielsweise Werkzeuganpassungen beim Zulieferer in der Entwicklung und Produktion oder weitere zeitaufwendige Bauteileausbauten und Konstruktionsänderungen.

Die Einführung updatefähiger Steuergeräte verursacht jedoch auch zusätzliche Aufwendungen. Es muss beispielsweise berücksichtigt werden, dass im Rahmen einer erhöhten Komplexität durch das Zusammenspiel der verschiedenen Komponenten Steuergerät, Funktion und System eine Änderung immer Aktualisierungen mehrerer Komponenten nach sich ziehen kann. Diese hohe Komplexität verlangt eine sorgfältige und durchgängige Konzept- und Systementwicklung der Steuergeräte- und Softwarearchitektur. Es müssen Schnittstellen und Methoden entsprechend definiert und die Kompatibilität innerhalb der Systemvernetzung gewährleistet werden. Zusätzlich kann nur durch die Gestaltung eines einheitlichen durchgängigen Gesamtprozesses entlang des Produktlebenszyklus eine problemlose Update-Programmierung in allen Funktionsbereichen ermöglicht werden.

Des Weiteren befürchten Fachleute, dass auf Grund der Möglichkeit zu beliebig häufigen Softwareänderungen die Gefahr einer unnötig überhöhten Änderungshäufigkeit z.B. durch geringere Arbeitssorgfalt oder schlecht durchdachter Lösungskonzepte besteht [Tauscher04]. Als Folge kann sich ein Qualitätsverlust bei

[22] Die Darstellung bezieht sich dabei nicht ausschließlich auf den Entwicklungsbereich, jedoch betrifft der überwiegende Teil der Zeiteinsparungen diesen Bereich.

der Softwarevalidierung einstellen oder unter dem Aspekt des hohen Zeitdrucks auch die Gefahr einer verfrühten Funktionsfreigabe erhöhen.

4.2.2 Beschaffung

Innerhalb der Beschaffung lassen sich Kostensenkungen durch den Einsatz von updatefähigen Steuergeräten erreichen, beispielsweise über einen niedrigeren Einkaufspreis pro Steuergerät oder einen effizienteren Beschaffungsprozess. Beides ist vornehmlich auf die Reduzierung der Variantenvielfalt der Steuergeräte zurück zuführen. So geben die Steuergerätezulieferer ihre Kostenersparnisse durch den Einsatz der Flash-Programmierung über den Preis an den Fahrzeughersteller weiter. Dieser spart zusätzlich Kosten mit einem reduzierten Beschaffungsaufwand durch ein vereinfachtes Varianten- und Zulieferermanagement. Weiter besteht ein großer Vorteil bezüglich der kurzfristigen Verfügbarkeit von Flash-Speicher gegenüber Masken-ROMs und damit der Verfügbarkeit von updatefähigen Steuergeräten gegenüber Steuergeräten mit Masken-ROM. Bei dem Flash-Speicher handelt es sich um ein verfügbares kundenunspezifisches Standardprodukt, während die Lieferzeit für den Masken-ROM auf Grund der kundenspezifischen Anpassungen im Vergleich dazu um einige Wochen länger sein kann. Dies senkt die benötigten Sicherheitsreserven des Lagerbestands, da weniger Steuergeräte für einen Lieferengpass vorgehalten werden müssen.

4.2.3 Logistik

Die Reduzierung der Variantenvielfalt hat eine vereinfachte Lagerhaltung mit geringeren Verwaltungsaufwand und Kommissionierungskosten, d.h. Kosten für die Zusammenstellung von Fahrzeugkomponenten entsprechend eines Auftrags, zur Folge und wirkt sich deshalb ebenfalls kostensenkend auf den Logistikbereich aus. Darüber hinaus sinken auf Grund geringerer notwendiger Lagerbestände die Kapitalbindungskosten. Dem gegenüber steht ein erhöhter Aufwand für das Versions- und Variantenmanagement der Software. Für diesen Zweck ist ein umfassendes und gepflegtes Dokumentationssystem notwendig. Dazu sind neben den Teilenummern für jede Steuergeräte-Hardware nunmehr auch Teilenummer für die einzelnen Softwareversionen notwendig.

4.2.4 Produktion

Durch die begrenzten Datenübertragungsraten derzeitiger Kraftfahrzeugdatenbusse und Diagnoseschnittstellen sind die momentanen Flash-Programmierzeiten bei großen Datenumfängen recht hoch. Eine standardmäßige Programmierung in der Linienproduktion am Bandende wird derzeit nur sehr vereinzelt durchgeführt. Durch den vorgegebenen, engen zeitlichen Rahmen der Produktion ist eine vollständige Programmierung des Speichers innerhalb der Fahrzeugproduktion nicht effizient. Daher sind die Nutzenpotentiale insgesamt geringer als in den Bereichen Logistik oder Kundendienst und Service. Wenn programmiert wird, dann in einem begrenzten Umfang, z.B. in der Vormontage oder für Nach- oder Umarbeiten, in der Vorserienproduktion oder bei Kleinserien- und Spezialfahrzeugen, wobei hier der Anteil bei den Nutzfahrzeugherstellern höher liegt als bei den Personen-

kraftwagenherstellern [Bernhart04]. So lassen sich durch die Update-Programmierung beispielsweise Zeit und Kosten beim Probeverbau der Steuergeräte in der Vorserienproduktion oder zur Korrektur des Einbaus eines falschen Steuergeräts in der Nachbearbeitung reduzieren.

4.2.5 Kundendienst und Service

Beim Kundendienst und Service steht das Aktions- und Variantenflashen im Vordergrund. Hierbei kann die Software entsprechend zur Fehlerbeseitigung und Funktionsanpassung und -erweiterung programmiert werden, ohne das Steuergerät zeit- und kostenaufwendig ausbauen und austauschen zu müssen. Damit lässt sich das Fahrzeug auch nachträglich mit neuer, verbesserter Software versorgen. Damit können Kosteneinsparungen durch Reduzierungen der Mobilitätsgarantieleistungen, der Gewährleistungskosten und des Ersatzteillagers erzielt werden.

Gleichzeitig besteht die Möglichkeit des nachträglichen Variantenmanagements, indem beispielsweise eine vom Hersteller autorisierte leistungssteigernde Anpassung der Motorkennlinie gegen Bezahlung vorgenommen werden kann. Die Update-Programmierung wird in diesem Bereich als Basis zur Generierung neuer Geschäftsfelder gewertet, indem zukünftig die Software als ein eigenständiges, vom Kunden zu erwerbendes, Produkt für zusätzlichen Umsatz sorgt. Das größte Potenzial für diese Vision wird in den Bereichen Antriebstrang und Multimedia gesehen [Huber03]. Eine Untersuchung ergab, dass bei den Kraftfahrzeugherstellern im Schnitt eine Erhöhung des zusätzlichen Umsatzpotenzials durch den nachträglichen Softwareverkauf von 3 % (2003) auf 13 % (2006) erwartet wird. Dies würde bei einem PKW einen geschätzten zusätzlichen Umsatz von ca. 200 € pro Fahrzeug bedeuten [Bernhart04].

4.2.6 Bewertung und Ausblick

Die Betrachtungen der Update-Programmierung entlang der Wertschöpfungskette zeigen, dass neben den rein technischen Anforderungen eine durchdachte Gestaltung des gesamten Flashprozesses über die verschiedenen Unternehmensbereiche eine entscheidende Rolle für den erfolgreichen Einsatz von Flash-Speicher spielt. Nur mit Hilfe durchgängig abgestimmter und gelebter Prozesse, beispielsweise bezüglich der Softwarevarianten, zwischen den Unternehmensbereichen wie Entwicklung, Produktion und Kundendienst kann ein fehlerfreier und einheitlicher Softwarestand im Fahrzeug gewährleistet werden.

Die wichtigsten Motive für die Einführung von Flash-Speichern mit der Möglichkeit der Update-Programmierung zeigt Abbildung 30, die einer Studie zu diesem Thema entnommen wurde [Bernhart04]. Daraus lässt sich ableiten, dass sowohl bei den Automobil- als auch bei den Nutzfahrzeugherstellern die erwartete Kostenreduzierung das Hauptmotiv für den Einsatz von Flash-Speicher darstellt. Weitere wichtige Punkte sind die Reduktion der Variantenvielfalt und die bessere Anpassung an die kurzen Innovationszyklen mit den daraus resultierenden hohen Änderungshäufigkeiten.

Die Aspekte Sicherung des geistigen Eigentums (IP, engl. Intellectual Property) und Wettbewerbsdifferenzierung werden bei den PKW-Herstellern unterschiedlich bewertet, was in einer sehr gleichmäßigen Verteilung über die einzelnen Präferenzen deutlich wird. Einige der Nutzfahrzeughersteller entwickeln ihre Steuergerätesoftware selbst, was den höheren Stellenwert als bei den PKW-Herstellern erklärt.

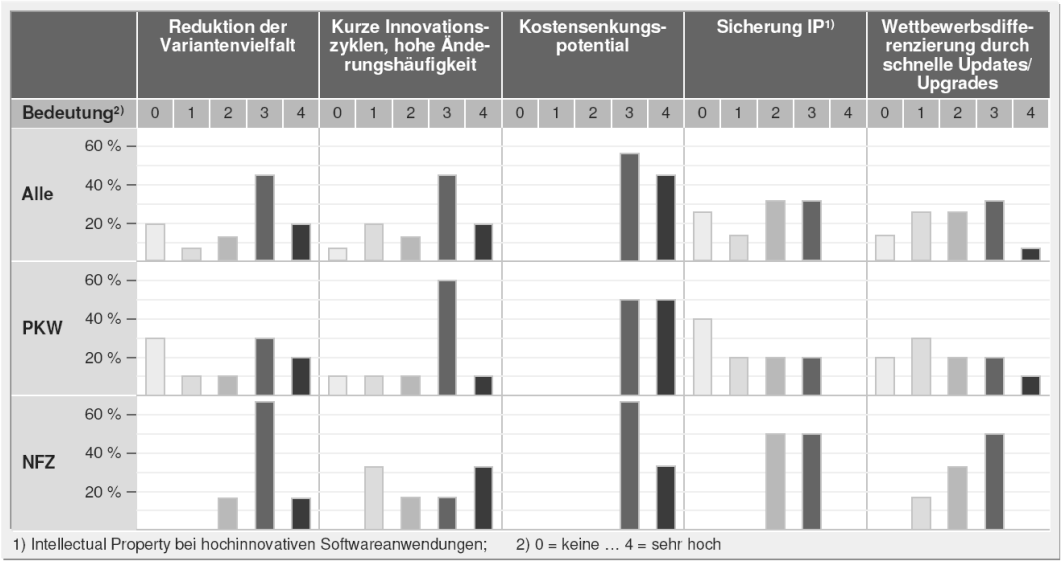

Abbildung 30: Hauptmotive für die Einsatz von Flash-Speichern im Kfz [Bernhart04]

In Abbildung 31 ist das prozentuale Kostensenkungspotential verschiedener Kosten für das Jahr 2003 sowie eine Prognose für das Jahr 2006 näher dargestellt. Das größte Einsparungspotenzial wird dabei, derzeit und zukünftig im Servicebereich durch die Reduzierung der Garantie- und Kulanzkosten gesehen.

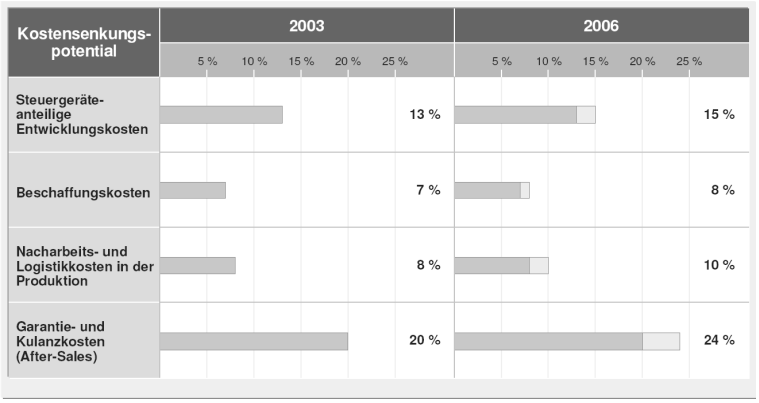

Abbildung 31: Kostensenkungspotenzial durch Flash-Speicher [Bernhart04]

Zusätzlich wurde das Flashen im Hinblick auf die verschiedenen Fahrzeugsubsysteme betrachtet (Abbildung 32). Dabei ist angegeben, wie hoch der prozentuale Anteil der programmierbaren Steuergeräte in einem Fahrzeugsubsystem ist oder zukünftig sein wird. Der höchste Anteil liegt hier im Bereich des Antriebsstrangs, der geringste im Bereich der Karosserie. Dies lässt sich damit begründen, dass ein hoher Anteil der Steuergeräte in der Karosseriedomäne über eine, im Vergleich zu

Steuergeräte aus dem Antriebsstrang, geringe Komplexität verfügen und deren Funktionsvariationen einen geringen zusätzlichen Kundennutzen darstellen. Beispiele für solche Steuergeräte aus dem Karosseriebereich sind Tür- oder Beleuchtungssteuergeräte, deren Änderungshäufigkeit relativ gering ist, so dass in diesem Fall der Einsatz von Masken-ROM wirtschaftlicher sein kann.

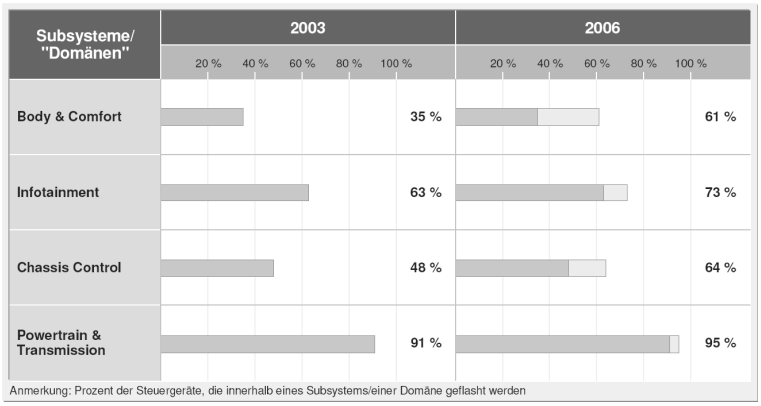

Abbildung 32: Erwarteter Flash-Bedarf der Domänen bei PKWs [Bernhart04]

Allerdings zeigt die Abbildung 32 auch, dass im Bereich der Karosserie der höchste Zuwachs an programmierbaren Steuergeräten erwartet wird. Dieser Zuwachs lässt sich mit der steigenden Komplexität im Bereich der Karosserie durch zusätzliche elektronische Systeme (z.B. gefordert durch Fußgängerschutz usw.) und dem prognostizierten Anstieg der Kosten für Masken-ROM erklären.

Die hier dargestellten Aspekte wurden bei [Torney04] für Flash-Speicher heraus gearbeitet und anschließend mit denen von Masken-ROM verglichen. Auf die Wiedergabe der einzelnen Details dieser Untersuchung wird hier verzichtet und allein das resultierende Ergebnis, das auf Grundlage einer wirtschafts-wissenschaftlichen Kosten-Nutzen-Rechnung entstand, vorgestellt. Die Ergebnisse sind in Tabelle 5 dargestellt, wobei ein mit ‚X' gekennzeichneter Bereich den Kosten bzw. den Erlös der jeweiligen Technologie darstellt. Es ist deutlich zu sehen, dass ein Masken-ROM über den gesamten Entwicklungszyklus nur Kosten verursacht, wogegen der Flash-Speicher auf Grund seiner Eigenschaften der Reprogrammierung im Feld neue Geschäftsfelder eröffnet. Als Ergebnis sind die Vorteile des Flash-Speichers im Vergleich zum Masken-ROM ersichtlich, so dass von einer Zunahme programmierbarer Steuergeräte im Fahrzeug auszugehen ist. Damit eng verbunden ist der Prozess der Update-Programmierung, der in im stärkeren Maße eingesetzt werden und dessen Reifegrad wesentlich zur Realisierung der Kostensenkungen durch die Flash-Speicher beitragen wird. Des Weiteren wird die Datenmenge, die über die Diagnoseschnittstelle in das Fahrzeug gebracht werden muss, ebenfalls stark zunehmen. Bei der Auslegung der Diagnoseschnittstelle und dem Entwurf von Kommunikationsarchitekturen zukünftiger Fahrzeuge muss diesem Umstand Rechnung getragen werden.

		Flash-Speicher		Masken-ROM	
		Kosten	Erlös	Kosten	Erlös
Entwicklung	Änderungszeit			X	
	Systementwurf	X			
	Änderungshäufigkeit	X			
Beschaffung	Einkaufspreis			X	
	Variantenmanagement			X	
	Zuliefererermanagement			X	
	Lieferzeiten			X	
Logistik	Kommissionierungskosten			X	
	Verwaltungsaufwand			X	
	Kapitalbindungskosten			X	
	Variantenmanagement			X	
Produktion	Umarbeitungskosten			X	
	Nacharbeitungskosten			X	
Kundendienst Service	Software als Produkt		X		
	Garantiekosten			X	
	Ersatzteillagerung			X	

Tabelle 5: Einteilung der Kosten- und Erlösstruktur anhand der Funktionsbereiche

4.3 Arten der Programmierung

Der Prozess der Programmierung, das „Flashen“, kann in verschiedene Arten unterteilt werden [Bernhart04]. Diese unterscheiden sich im Wesentlichen durch den Umfang der zu programmierenden Daten und den unterschiedlichen Auswirkungen auf die Steuergerätesoftware mit ihren Funktionen. Der technische Vorgang der Programmierung, das eigentliche Beschreiben des Flash-Speichers, ist bei allen Arten gleich. Für diese Arbeit ist das Flashen auf Grund der hohen Datenmengen interessant.

4.3.1 Flashen

Der Begriff „Flashen“ wird meist als Oberbegriff für einen allgemeinen Programmierungvorgang in den Flash-Speicher eines Steuergeräts benutzt. Meist ist damit das Aufspielen von Software-Komponenten gemeint, die beispielsweise Daten, funktionale oder Diagnoseprogramme, Bootsoftware und Betriebssysteme sein können. Die Programmierung derartiger Komponenten kann zusätzlich oder im Austausch zu bereits vorinstallierten Modulen auf einem Steuergerät erfolgen. Über das Flashen wird meist ein sehr großer Teil des verfügbaren Flash-Speichers eines zu programmierenden Steuergeräts beschrieben. Die Datenmengen können sich zwischen einigen Kilobytes bis hin zu mehreren Megabytes bewegen.

4.3.2 Parametrieren

Beim Parametrieren, auch als Anpassung bezeichnet, handelt es sich um Lern- und Adaptionsvorgänge in den Steuergeräten während des Produktionsdurchlaufes. Hier wird zumeist ein abgegrenzter Speicherbereich mit einem vorher ermittelten Wert beschrieben. Bei dieser Art der Programmierung werden nur einige Bytes transportiert und in den Flash-Speicher geschrieben. Die eigentliche Funktionssoftware im Steuergerät bleibt unverändert. Daher ist diese Art der schnellen Programmierung relativ unkritisch gegenüber Störungen von außen.

4.3.3 Kodieren

Als Kodieren wird die Anpassung der Steuergeräte an die Peripherie bezeichnet, welche abhängig von der Fahrzeugausstattung ist. Bei dieser Aktivierung von Fahrzeugbesonderheiten erfolgt kein Eingriff in die Software des Steuergerätes. Hierbei werden lediglich einzelne Bits gesetzt, die bestimmte Fahrzeugfunktionalitäten aktivieren oder deaktivieren. Einsatzmöglichkeiten für das Kodieren sind die Band-Ende-Kodierung für ein Steuergerät, bei der über ein zu setzendes Bit beispielsweise die Art des Getriebes (Schalt- oder Automatikgetriebe) kodiert wird oder die Freischaltung vorhandener Funktionen auf dem Steuergerät.

4.4 Kommunikation zur Update-Programmierung

4.4.1 Ablauf einer Update-Programmierung

Mit Hilfe eines geeigneten Werkzeugs bzw. Programmiergeräts (Diagnosetester), das über die Diagnoseschnittstelle mit dem vernetzten Steuergerät kommunizieren kann, gelangt die aktuelle Software über die Fahrzeugbusleitung zum Steuergerät. Die dafür notwendigen Protokolle wie Transport- und Diagnoseprotokolle werden anschließend beschrieben. Der große Vorteil liegt hierbei in der Möglichkeit zum Software-Update ohne Steuergeräteausbau, was zu erheblichen Zeit- und Kosteneinsparungen führen kann. Jeder Fahrzeughersteller verfolgt seine eigene Strategie für den Ablauf der Update-Programmierung, so dass dieser derzeit noch von Hersteller zu Hersteller bezüglich Authentifizierung, Sicherheit und Programmierungsprozess unterschiedlich realisiert sein kann [Tauscher04]. Daher erfolgt hier eine Beschränkung auf den prinzipiellen Ablauf, der in Abbildung 33 dargestellt und bei den meisten Fahrzeugherstellern ähnlich ist. Der Ablauf einer Update-Programmierung lässt sich grob in fünf Hauptphasen unterteilen.

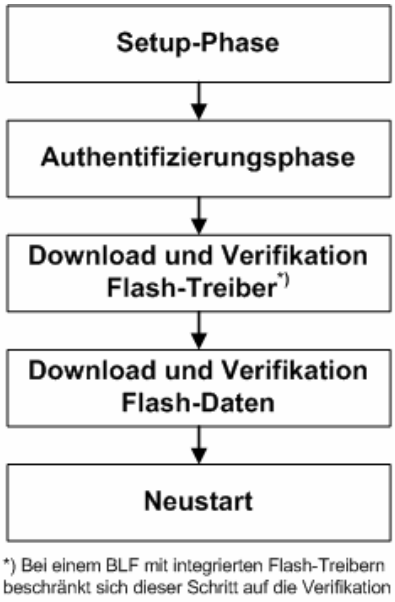

Abbildung 33: Prinzipieller Ablauf einer Update-Programmierung

4.4.1.1 Setup-Phase

Innerhalb der Setup-Phase wird die benötigte Hardware initialisiert sowie die externe Programmieranforderung und die Gültigkeit des Flashloaders überprüft. Beispielsweise vergleicht das Programmiergerät die vorhandene Softwareversion des Steuergeräts mit der zu programmierenden Version, so dass nur die Programmierung aktuellerer Softwareversionen erlaubt wird. Für diese Schritte stellen standardisierte Diagnoseprotokolle geeignete Mechanismen bereit. Zur Kommunikation greifen diese Diagnoseprotokolle auf Transportprotokolle zurück.

4.4.1.2 Authentifizierungsphase

Während der Authentifizierungsphase muss sichergestellt werden, dass das Steuergerät nur von autorisierten Personen und Programmiergeräten programmiert werden kann. Aus diesem Grund wird der Programmierzugriff auf den Flash-Speicher über eine Authentisierung des Programmiergeräts, die über spezielle herstellerspezifische Mechanismen durchgeführt werden, abgesichert.

4.4.1.3 Download und Verifikation der Flash-Treiber

Ist die Authentifizierung erfolgreich, wird der Bootloader-Flash (BLF) gestartet. Beinhaltet der BLF die Flash-Routinen, kann sofort mit dem Download der Flash-Daten begonnen werden. Zur weiteren Erhöhung der Sicherheit besteht die Möglichkeit, die zum Löschen und Programmieren benötigten Flash-Routinen nicht auf dem Steuergerät zu speichern, sondern nach erfolgreicher Authentifizierung vom Programmiergerät auf das Steuergerät herunter zuladen. Erst nach erfolgreicher Überprüfung der Korrektheit und Freigabe des Flash-Treibers, der vom Kraftfahrzeughersteller zertifiziert wird und ohne den eine Programmierung nicht möglich ist, kann dieser verwendet werden.

4.4.1.4 Download und Verifikation der Flash-Daten

Nach erfolgreicher Löschung der entsprechenden Speicherblöcke oder des gesamten Flash-Speichers (außer dem Bereich des BLF) werden die neu zu programmierenden Daten übertragen. Es muss sichergestellt sein, dass es sich um die vom Hersteller zertifizierten und zum Steuergerät passenden Daten handelt, um Fehlfunktionen oder einem Missbrauch vorzubeugen. In einem weiteren Prüfschritt wird die Datenkonsistenz mit Hilfe von Verschlüsselungsverfahren und Checksummen überprüft, um unzulässige Manipulationen, fehlerhafte Software und Fehler bei der Übertragung oder der Programmierung in den Flash-Speicher auszuschließen. Diese Prüfungen können sich entsprechend der Sicherheitsansprüche der verschiedenen Hersteller unterscheiden.

4.4.1.5 Neustart

Abschließend werden die Verwaltungsinformationen aktualisiert und die Flash-Routinen aus dem RAM gelöscht, indem das Steuergerät neu gestartet wird und mit der neuen Softwareversion arbeitet.

4.4.2 Transportprotokolle

Zur Übertragung von größeren Datenmengen wie sie beispielsweise bei der Update-Programmierung auftreten, kommen Transportprotokolle zum Einsatz. Diese Protokolle arbeiten auf der Schicht 4 des ISO/OSI-7-Schichtenmodells, die als Transportschicht bezeichnet wird. Ihre Aufgabe ist es, Daten, deren Größe die Protocol Data Unit (PDU)[23] des Übertragungskanals überschreitet, beim Sender geeignet zu segmentieren und nach der Übertragung beim Empfänger in der richtigen Reihenfolge wieder zusammenzusetzen. Dazu kümmert sich das Transportprotokoll um den Auf- und Abbau des Übertragungskanals und übernimmt die Flusskontrolle für diesen Kanal. Die Art der Daten spielen für die Übertragung auf der Transportschicht keine Rolle. Über der Transportschicht setzen vielmehr weitere Applikationsprotokolle wie beispielsweise Diagnoseprotokolle (KWP, UDS) oder Anzeigeprotokolle (DDP) auf, die den Transportkanal nutzen und die Daten geeignet interpretieren. Der große Vorteil eines Transportprotokolls liegt hier in der Entkopplung der Applikation vom eingesetzten Datenbussystem.

Die derzeit in der Automobiltechnik eingesetzten Transportprotokolle setzen historisch bedingt auf dem CAN-Bus auf, der eine maximale PDU von 8 Byte anbietet. Eines der ersten Transportprotokolle für den CAN-Datenbus ist das MC-Net der Firma Bosch. Volkswagen leitete daraus sein eigenes Transportprotokoll der Version 1.6 (TP 1.6) ab, welches noch bis in die Modellreihen des Golfs der 4. Generation eingesetzt wurde. Das Transportprotokoll TP 1.6 wurde von Volkswagen zum TP 2.0 weiterentwickelt und wird derzeit im gesamten Volkswagen-Konzern neben TP 1.6 eingesetzt [TP2.0]. Eine allgemeine Version eines Transportprotokolls ist als ISO-TP (ISO-Norm 15765, Teil 2) standardisiert und wird von vielen Fahrzeugherstellern verwendet [ISO15765].

[23] Größte Datenmenge, die zusammenhängend in einen Nachrichtenrahmen passt.

Langfristig wird sich ISO-TP als offener Standard gegenüber TP 2.0 durchsetzen. Allerdings sind sich die beiden Protokolle in vielen Bereichen sehr ähnlich. Die wichtigsten Unterschiede sind die Möglichkeit der Fehlerbehandlung bei TP 2.0, das dadurch im Fehlerfall den Übertragungskanal effizienter nutzen kann sowie die Eigenschaft des TP 2.0 einen Übertragungskanal auch ohne aktive Übertragung offen zu halten. Das ISO-TP dagegen besitzt einen verkürzten Kanalaufbau, da bei diesem Protokoll die Übertragungsidentifier vorab festgelegt sind. Hier bietet TP 2.0 im Gegensatz die Möglichkeit, über den Kanalaufbau auf einer ebenfalls festgelegten ID die späteren Übertragungs-IDs mitzuteilen, was in der Praxis allerdings kaum Vorteile bringt. Auf Grund dessen, dass sich beide Protokolle in der reinen Datenübertragung ohne Kanalaufbau und –abbau kaum unterscheiden und diese Arbeit in enger Zusammenarbeit mit der Volkswagen AG entstand, wurde als Transportprotokoll TP 2.0 anstelle des ISO-TP eingesetzt.

Die hier vorgestellten Transportprotokolle wie TP 2.0 und ISO-TP sind speziell für den CAN-Bus konzipiert. Für andere Datenbusse als dem CAN-Bus müssen weitere Transportprotokolle eingesetzt werden, die sich allerdings am Aufbau des ISO-TP orientieren können. Für FlexRay beispielsweise existiert noch kein Transport-protokoll, so dass im Rahmen der Arbeit ein eigener Vorschlag entwickelt und implementiert wurde. Mittelfristig ist davon auszugehen, dass über das AUTOSAR-Konsortium derartige benötigte Protokolle für das Kraftfahrzeug spezifiziert werden. Das hier vorgestellte Transport- und Routingprotokoll kann dazu als Beispiel und Anregung dienen. Eine weitere Einschränkung bezüglich der Transportprotokolle liegt in der fehlenden Vermittlungsschicht. Damit ist mit den derzeitigen Mechanismen keine Datenbus-übergreifende Adressierung von Steuergeräten möglich. Für eine dezentrale Architektur wie der Backbone-Architektur ist dies jedoch eine essentielle Voraussetzung. Im Rahmen dieser Arbeit wurde aus diesem Grund ein Vermittlungsprotokoll unterhalb der Transportprotokolle entwickelt, das in Abschnitt 6.3 vorgestellt wird. Schwerpunkt dieses Protokolls bildet die netzwerkweit eindeutige Adressierung, deren Entwicklung einen guten Kompromiss aus Flexibilität und Übertragungseffizienz darstellt. Das Vermittlungsprotokoll kann als Vorschlag für dezentrale Netzwerke dienen, wie AUTOSAR sie propagiert. Die in der Arbeit gewonnenen Erkenntnisse bezüglich Übertragungsverhalten und Leistungsfähigkeit liefern dabei wichtige Ansatzpunkte.

4.4.3 Diagnoseprotokolle

Die im Kraftfahrzeug eingesetzten Diagnoseprotokolle stellen aus Sicht des ISO/OSI-7-Schichtenmodells eine Anwendung dar und arbeiten auf der Schicht 7, der Applikationsschicht. Sie definieren Dienste und entsprechende Schnittstellen, um Informationen aus dem Steuergerät abzufragen (z.B. Fehlerspeichereinträge, Teile-nummern, Messwerte), dem Steuergerät Aktionsanweisungen zu geben (z.B. Fehler-speicher löschen, Stellgliedtest) und Daten dem Steuergerät zu übergeben (z.B. Download Flashdaten, Kodierung, Anpassung). Die verwendeten Diagnoseprotokolle ermöglichen über ihre Dienste und die eingesetzten Transportprotokolle eine Skalierung des Programmier-Ablaufs, angepasst an die Leistungsfähigkeit des Steuergeräts oder seiner Sicherheitsklasse. Die im Einzelnen vorgestellten Diagnose-protokolle sind in ihrer historischen Entwicklung in Abbildung 34 zusammengefasst dargestellt.

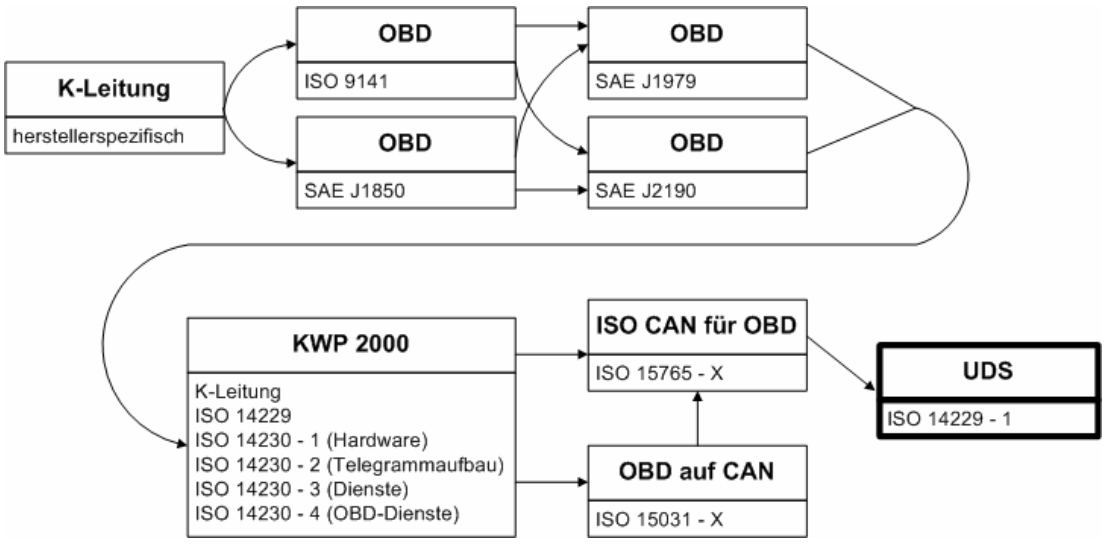

Abbildung 34: Überblick über die Historie der Diagnoseprotokolle [Ellinger04]

Als Vorläufer der Diagnoseprotokolle kann der Blinkcode gelten, der speziell für die Diagnose des Motors entwickelt wurde. Dazu war im Motorraum eines Fahrzeugs ein Diagnosestecker mit einem K-Pin und einem (optionalen) L-Pin vorhanden [Zimmerm.06]. Durch das „auf Masse schalten“ des L-Pins für eine bestimmte Zeitdauer (mind. 2,86 s) wurde die Diagnose gestartet. Die Ausgabe erfolgte mittels vierstelliger Zahlen über den K-Pin, der an eine Lampe angeschlossen war, die entsprechend der Ziffern, nur eins bis vier waren erlaubt, blinkte. Über eine Tabelle konnte diese Information des Blinkcodes mit einem Fehlereintrag verknüpft werden. Diese Blinkcodes werden auch heute noch in den Fehlercodes mitgeführt. Allerdings eignete sich der Blinkcode nur für eine sehr einfache Diagnose mit begrenzter Aussage. Dazu bedurfte es immer eines Mechanikers, der die Blinkcodes mitzählte und auswertete. Daher wurde dieses Vorgehen bald durch leistungsfähigere Diagnoseprotokolle abgelöst.

4.4.3.1 Keyword-Protokoll 1281 (KWP 1281)

Die Firma BOSCH entwickelte ein solches leistungsfähigeres Diagnoseprotokoll für ihre Motorsteuergeräte unter dem Namen Keyword-Protokoll 71. Dieses Protokoll wurde sehr schnell von den Automobilherstellern adaptiert und eingesetzt. Allerdings sind keine der einzelnen Kommunikationsprotokollroutinen der verschiedenen Hersteller miteinander kompatibel. Ein Diagnosetester, der mit einem Fahrzeug kommunizieren wollte, musste daher den „Dialekt“ der jeweiligen KWP-Ausprägung erfragen. Dies geschah über sogenannte Keywords, von denen sich der Name des Protokolls ableitete. Das im Volkswagen-Konzern eingesetzte Keyword-Protokoll wurde als KWP 1281 bezeichnet.

KWP 1281 wird über die K-Leitung als serielles Protokoll eingesetzt und kann mittlerweile auch über CAN transportiert werden (TP 1.6). Es nutzt eine 5 Baud-reizung, auf die das adressierte Steuergerät mit einem Synchronisationsbyte und zwei Keybytes antwortet. Durch die vom Steuergerät mitgeschickten Keybytes wird das zu benutzende Keyword-Protokoll bestimmt. Mit Hilfe des Synchronisationsbytes kann der Tester die Baudrate des Steuergeräts ermitteln, die nicht im Protokoll

festgelegt ist. Allerdings haben sich mit der Zeit 9,6 kBaud und 10,4 kBaud (herstellerabhängig) als Quasi-Standard-Baudrate etabliert.

4.4.3.2 On-Board-Diagnose II (OBD II)

Die On-Board-Diagnose II (OBD II) ist in den USA ein Gesetz für alle Fahrzeuge, die Abgase verursachen. Sie stellt eine Eigendiagnose für den Motor und andere abgasrelevante Komponenten dar. Bei einer Fehlfunktion muss diese dem Fahrer mitgeteilt werden (MIL, engl. Malfunction Indicator Lamp). Das Problem der verschiedenen inkompatiblen Diagnoseprotokolle der einzelnen Hersteller wurde mittels OBD II gelöst, in dem zwingend eine einheitliche Diagnosekommunikation vorgeschrieben wurde. OBD II ist in den Normen ISO 9141, ISO 14230-4 und ISO/CD 15031 definiert.

Die OBD II-Norm nutzt zwei Bytes für die einzelnen Fehlercodes. Diese sind in vier große Fehlerbereiche unterteilt, die als P-Codes (Powertrain), C-Codes (Chassis), B-Codes (Body) und U-Codes (Network) bezeichnet werden. Für die Hersteller sind spezielle Bereiche innerhalb der Codes eingeräumt, die für eigene Zwecke genutzt werden können. Als wesentliche Neuerung wird bei OBD II die funktionale Adressierung eingeführt, bei der nicht mehr ein einzelnes Steuergerät, sondern eine Funktion abgefragt wird. Alle an der Funktion beteiligten Steuergeräte antworten nacheinander auf die Anfrage.

4.4.3.3 Keyword-Protokoll 2000 (KWP 2000)

Durch die Notwendigkeit der OBD II-Norm, die für abgasrelevante Diagnosedaten zuständig ist, lag es nahe, auch alle anderen Diagnosedaten in einem einheitlichen Diagnoseprotokoll zusammenzufassen. Somit wurde das Keyword-Protokoll 2000 (KWP2000) in der ISO 14230 als standardisiertes Diagnoseprotokoll für Steuergeräte entwickelt. Wobei die Standardisierung nicht wörtlich zu nehmen ist, da jeweils fahrzeugherstellerspezifische Ausprägungen existieren. Die OBD II-Norm ist inzwischen eine Untermenge von KWP 2000. Der physikalische Zugriff durch KWP 2000 kann über eine serielle Leitung ("K-Leitung") oder über CAN (KWP-on-CAN mittels Transportprotokoll) erfolgen.

4.4.3.4 Unified Diagnostic Service (UDS)

Das Diagnoseprotokoll KWP 2000 ist zwar standardisiert, es existieren jedoch eine Vielzahl unterschiedlicher Dialekte bei den einzelnen Herstellern. Durch die hohe Komplexität der einzelnen Fahrzeuge entstehen immer mehr Partnerschaften der Fahrzeughersteller untereinander. Hier zeigte sich, wie wichtig ein gemeinsames Diagnoseprotokoll für alle ist. Daher wurde das Unified Diagnostic Service-Protokoll (UDS) entwickelt, das es sich zur Aufgabe gemacht hat, sich als ein wirklich herstellerübergreifendes standardisiertes Diagnoseprotokoll zu etablieren. Es ist eine Weiterentwicklung von KWP 2000 und in der internationalen ISO 14229-1 Norm festgeschrieben. UDS nutzt auf dem CAN-Bus als Transportprotokoll ISO-TP.

Im Vergleich zu KWP 2000 sind die einzelnen Diagnosemodes tiefer verschachtelt und die Fehlerüberwachung durch weitere Überwachungsstatistiken und Auslese-

dienste erweitert. Zusätzliche Dienste, speziell zur Update-Programmierung, wie „Mute“ oder „Resetfunktion“ werden bei UDS eingeführt. Darüber hinaus setzt UDS auf Arbeiten des ASAM auf und nutzt zur Diagnose-Bedatung die dort festgeschriebenen ODX-Daten.

4.4.4 Systemintegrität

Eine der großen Herausforderungen, vor der die Automobilindustrie auf Grund der schnellen und relativ einfachen Austauschbarkeit der Softwarekomponenten steht, ist die Sicherstellung der Integrität des elektronischen Fahrzeug-Gesamtsystems. Während bisher die Software mit einem Steuergerät untrennbar verbunden war, zwingen die zunehmenden softwarebasierten Systeme im Fahrzeug und ihr modularer Aufbau die Automobilhersteller mehr und mehr, Software unabhängig von einer Hardware zu verwalten. Dazu gehören die Einführung einer Teilenummer, wie sie bislang ausschließlich für Hardwarekomponenten üblich waren und der Nachweis der Funktion für bestimmte Hardwarekonfigurationen. Solch ein Nachweis beinhaltet die vollständige Dokumentation und verlangt die Sicherstellung, dass nur freigegebene Kombinationen von Software und Hardware innerhalb eines Fahrzeugs zum Einsatz kommen. Auf diese Weise lässt sich bei neu produzierten Fahrzeugen die Integrität des elektronischen Gesamtsystems gewährleisten. Befindet sich das Fahrzeug allerdings erst in Kundenhand, lässt sich kaum noch prüfen, ob und in wie weit Änderungen am elektronischen Gesamtsystem vorgenommen worden sind bzw. welche nachträglichen Änderungen seitens des Kundendienstes erlaubt sind. Für diese Aufgabe sind zusätzliche Mechanismen und Verfahren notwendig, die eine Sicherstellung der Systemintegrität über den gesamten Lebenszyklus eines Kraftfahrzeugs ermöglichen. Einen Ansatz für solch ein Systemintegritätsprotokoll (SIP) wird in [TEhlers03] vorgestellt.

5 Aufbau eines heterogenen Kommunikationsnetzwerks

Ausgehend vom derzeitigen Entwicklungsstand der Kraftfahrzeugelektronik lassen sich Trends für zukünftige Entwicklungen ableiten. Der verbreitete Ansatz eines zentralen Super-Gateways im Kraftfahrzeug ist mit wachsendem Datenaufkommen, zunehmender Heterogenität automobiler Netzwerke und mit vertretbarem Aufwand kaum noch zu beherrschen. Allerdings ist sie etabliert und funktioniert, wobei die Leistungsfähigkeit für zukünftige automobile Anwendungen derzeit diskutiert wird [Vaßen05]. Die Backbone-Architektur verspricht die Nachteile der Gateway-Architektur zu beheben. Sie muss allerdings ihre Funktion und Vorteile erst unter Beweis stellen, was eines der Ergebnisse dieser Arbeit sein wird. Für die Untersuchungen ist daher der Zugriff auf ein Kraftfahrzeug-Kommunikationsnetzwerk, das auf einer solchen Backbone-Architektur beruht, wichtig. Da in verfügbaren Serienfahrzeugen die Gateway-Architektur vorherrschend ist, bestand die Notwendigkeit, im Labor ein prototypisches Kraftfahrzeug-Kommunikationsnetzwerk aufzubauen. Dieses Vorgehen und die dahinter stehenden Überlegungen werden in den weiteren Abschnitten dieses Kapitels beschrieben.

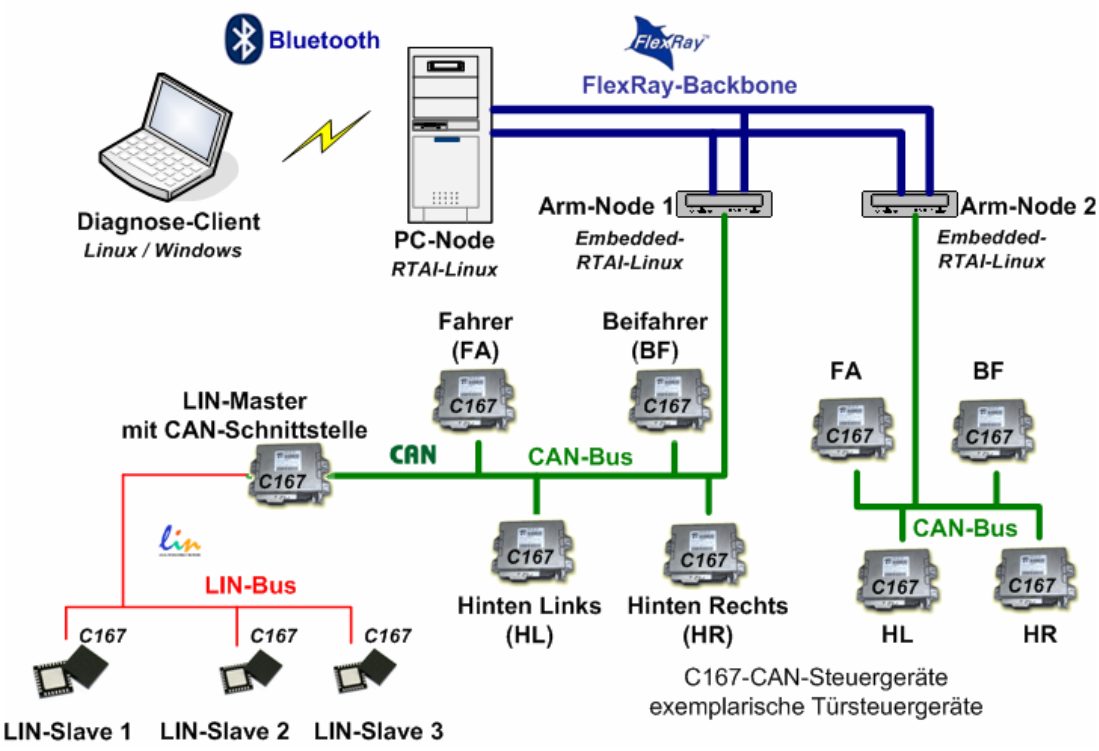

Abbildung 35: Labor-Kommunikationsnetzwerk mit FlexRay Backbone

5.1 Beschreibung der eingesetzten Hardware

Einen schematischen Überblick über das aufgebaute Kommunikationsnetzwerk und seinen einzelnen Teilnetzwerken gibt die Abbildung 35. In den folgenden Abschnitten werden die einzelnen Netzwerkkomponenten näher beschrieben.

5.1.1 FlexRay Backbone

Einen wesentlichen Bestandteil der zu untersuchenden Architektur bildet der Backbone-Datenbus, der im Rahmen dieser Arbeit mit FlexRay realisiert wurde. Die Überlegungen, die zum Einsatz des FlexRay-Datenbusses als Backbone führten, sind im Kapitel 3.5.3 beschrieben. Zum Zeitpunkt der Entstehung der Arbeit befand sich das FlexRay-Protokoll im Stadium der Spezifizierung. Aus diesem Grund wurden Prototypen von FlexRay-Controllern eingesetzt, die noch nicht über die volle Leistungsfähigkeit des FlexRay verfügen, aber die wesentlichen Eigenschaften des FlexRay schon sehr gut abbilden. Der verwendete FlexRay-Cluster der Firma Decomsys [Decomsys] besteht aus drei Prototypen-Steuergeräten (Knoten, engl. Nodes), die über einen 5 MBit/s schnellen FlexRay-Datenbus verbunden sind[24].

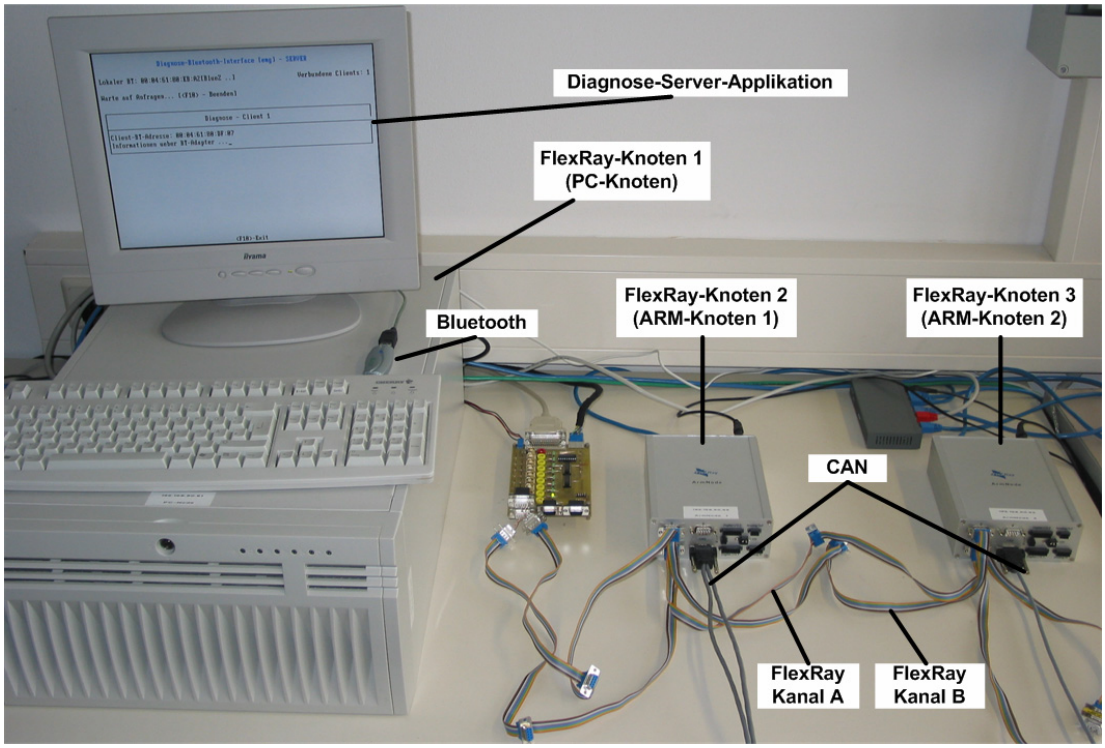

Abbildung 36: FlexRay-Cluster des Labornetzwerk

Da es sich bei FlexRay um einen zeitgesteuerten Datenbus handelt, verfügen die FlexRay-Steuergeräte über ein Echtzeitbetriebssystem. Zum Einsatz kommt dabei RTAI Linux [RTAI], das sowohl für das Standard Linux als auch für Embedded

[24] Die 5 MBit/s sind dem frühen Prototypen-Stadium geschuldet. Die FlexRay-Version, die später in Serie im Fahrzeug eingesetzt werden wird, verfügt über die in der Spezifikation festgelegten 10 MBit/s.

Linux verfügbar ist. Es erlaubt die Ausführung von Tasks mit harten Echtzeitbedingungen und bietet gleichzeitig den Zugriff auf alle Möglichkeiten, die Linux besitzt. Damit stehen Treiber für eine Vielzahl von Architekturen und Schnittstellen zur Verfügung und können für eigene Anwendungen verwendet werden.

5.1.1.1 ARM-Knoten

Zwei der drei Steuergeräte des FlexRay-Clusters basieren auf einem ARM 9-Controller. An diesem ist der FlexRay-Controller angeschlossen, der zu dem Zeitpunkt noch nicht als eigenständige Hardware existierte, sondern mittels eines FPGA realisiert wurde. Zusätzlich besitzen die ARM Knoten einen CAN-Controller, der zwei High-Speed CAN-Schnittstellen zur Verfügung stellt. Für Entwicklungs- und Debugzwecke sind an den Knoten eine Ethernet- sowie eine serielle Schnittstelle vorgesehen. Die ARM-Knoten laufen unter einem echtzeitgesteuerten Embedded Linux, für das Treiber zu den verschiedenen Schnittstellen inklusive FlexRay und CAN zur Verfügung stehen. Die Hauptaufgabe der ARM Knoten innerhalb der Kommunikationsarchitektur ist die Bereitstellung der Gatewayfunktionalität zwischen FlexRay-Backbone und CAN-Datenbus.

5.1.1.2 PC-Knoten

Bei dem dritten Steuergerät des FlexRay-Backbone handelt es sich im einen vollwertigen PC, der den Vorteil der einfachen Erweiterbarkeit mit handelsüblicher Hardware und Schnittstellen bietet. Der PC-Knoten verfügt über eine FlexRay-Schnittstelle, die per PCI-Bus an den PC angebunden ist. Ähnlich wie bei den ARM-Knoten sind die FlexRay-Controller als FPGA auf einer PCI-Karte integriert. Der PC-Knoten läuft im Gegensatz zu den ARM-Knoten unter einem echtzeitgesteuerten Standard Linux basierend auf Debian [Debian], für das neben den Linux-üblichen Treibern auch Treiber für die FlexRay-Hardware existieren. Die Aufgabe des PC-Knotens ist die Bereitstellung des Gateways für den drahtlosen Zugang nach außen. Zu diesem Zweck wurde über USB ein Bluetooth-Dongle integriert und mittels des freien Bluetooth Stack BlueZ [BlueZ] in das RTAI Linux eingebunden.

5.1.2 CAN-Netzwerke

Der CAN-Bus ist der derzeit am häufigsten eingesetzte Datenbus im Kraftfahrzeug. Aus diesem Grund sind für das Labor-Kommunikationssystem zwei CAN-Netzwerke entstanden, die über die ARM-Knoten mit dem FlexRay-Backbone verbunden sind. Beide CAN-Netzwerke sind nahezu identisch aufgebaut. Sie bestehen jeweils aus vier selbst entwickelten Steuergeräten auf Basis des Infineon C167-Mikrocontrollers. Dieser Mikrocontroller verfügt über einen CAN-Controller, mit dem sowohl der Anschluss an ein Low-Speed- als auch an ein High-Speed-CAN-Netzwerk möglich ist. Als exemplarische Applikation wurden auf den vier C167-Mikrocontrollern Türsteuergeräteapplikationen realisiert, bestehend aus Fahrer (FA), Beifahrer (BF), Hinten Links (HL) und Hinten Rechts (HR). Da die reale Funktion, die Ansteuerung eines Fensters bzw. einer Tür, für die Untersuchungen nur eine untergeordnete Rolle spielt und es vielmehr auf die Kommunikation der Steuergeräte ankommt, werden die Funktionen der Steuergeräte über ein LCD und verschiedene

LEDs visualisiert. Durch den in der Applikation vorgesehenen Datenaustausch der Steuergeräte untereinander wird dabei die Domänenkommunikation simuliert.

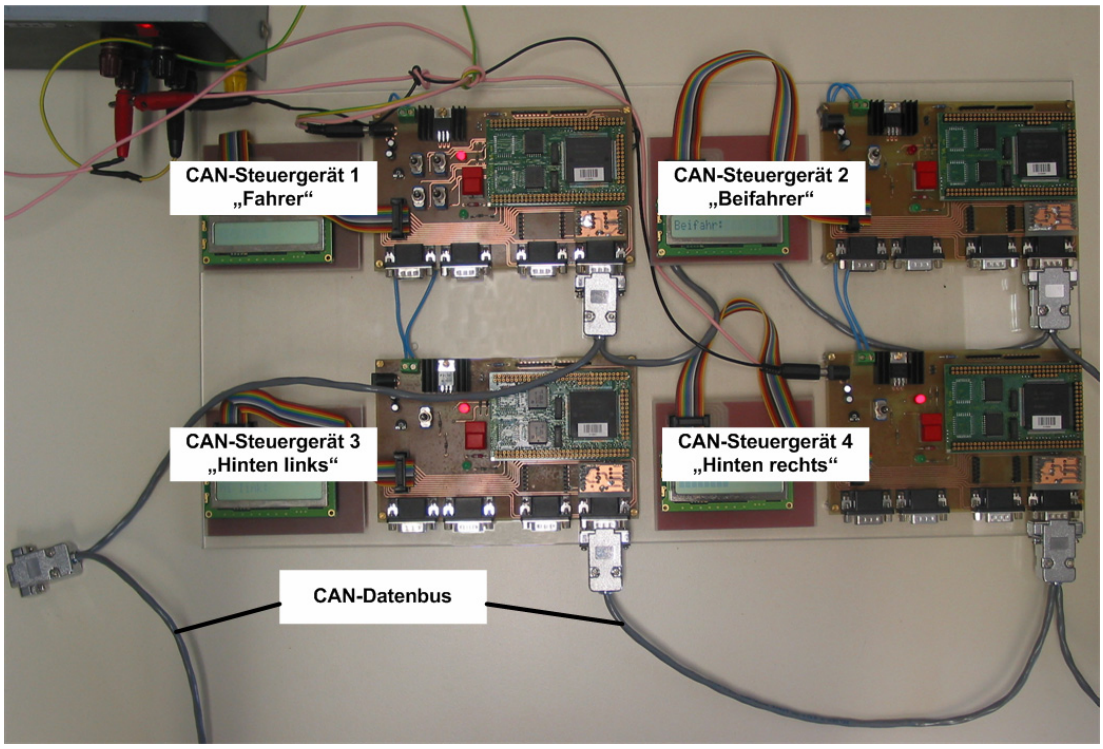

Abbildung 37: CAN-Aufbau des Labornetzwerks

Die Hauptaufgabe der CAN-Steuergeräte innerhalb des Labornetzwerks ist die Möglichkeit der Update-Programmierung über den CAN-Bus. Zu diesem Zweck existiert eine zweite, modifizierte, Version der Türsteuergeräte-Applikation, die geringfügig andere Reaktionen als die ursprüngliche Version produziert. Diese beiden Applikationsversionen können wechselseitig aus Sicht der Steuergeräte über den CAN-Bus geflasht werden. Aus Sicht des Kommunikationssystems werden die Steuergeräte von „außen“, über die drahtlose Diagnoseschnittstelle, programmiert. Aus Kostengründen konnte auf den C167 kein OSEK-Betriebssystem eingesetzt werden, so dass die entwickelten Applikationen die vollständige Verwaltung aller Ressourcen (z.B. Interruptbehandlung) mit übernehmen müssen.

5.1.3 LIN-Netzwerk

Ein ebenfalls im Kraftfahrzeug immer häufiger zum Einsatz kommender Datenbus ist der LIN-Datenbus. Da sich ein LIN-Netzwerk relativ kostengünstig realisieren lässt, wurde im Rahmen dieser Arbeit ein derartiges LIN-Netzwerk aufgebaut (siehe Abbildung 38). Über einen LIN-Bus sind in der Regel einfache Mikrocontroller, meist intelligente Sensoren und Aktoren, miteinander vernetzt. Um Kosten und Aufwand zu reduzieren, wurden die drei LIN-Slave-Steuergeräte und ein LIN-Master-Steuergerät für das Labornetzwerk ebenfalls auf Basis der C167-Mikrocontroller realisiert. Die CAN-Schnittstelle kommt bei den LIN-Slaves nicht zum Einsatz. Eine Ausnahme bildet der LIN-Master, der gleichzeitig mit der CAN-Schnittstelle das Gateway zu einem der CAN-Busse bildet.

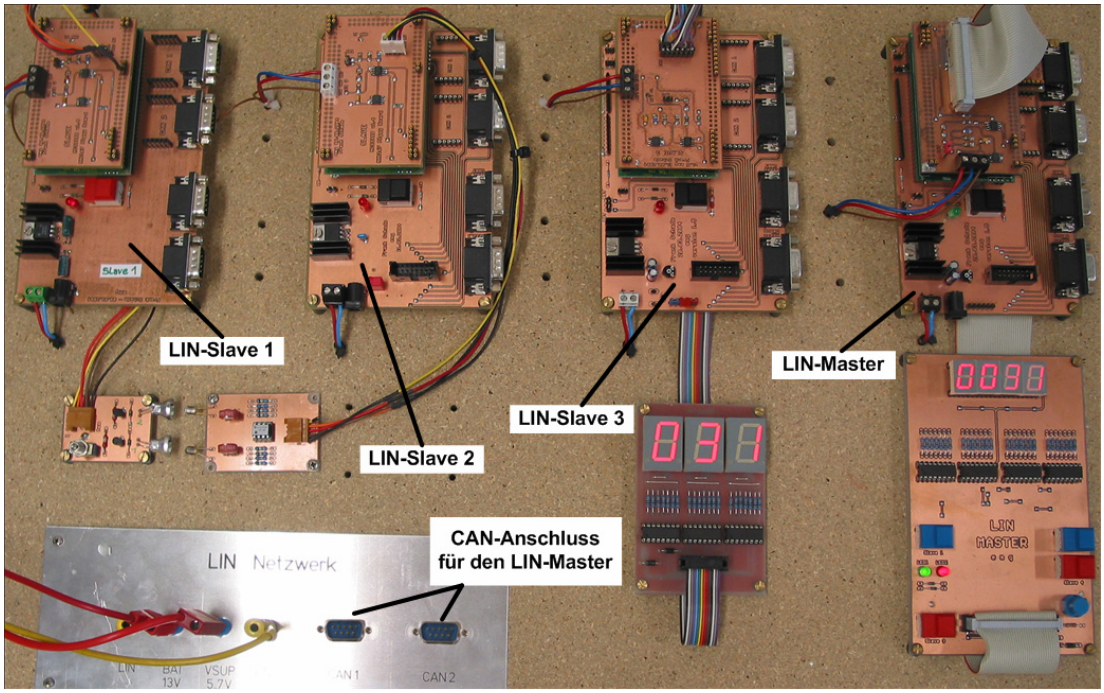

Abbildung 38: LIN-Aufbau des Labornetzwerks

Für jede dieser LIN-Steuergeräte existieren zwei Applikationsversionen, die sich in ihren Reaktionen geringfügig voneinander unterscheiden. Die einzelnen Applikationen der Steuergeräte benötigen für ihre Funktion Daten der anderen Steuergeräte, die über den LIN-Bus kommuniziert werden. Für diese Steuergeräte ist eine Update-Programmierung über den LIN-Bus möglich. Allerdings wird dafür ein CAN-LIN-Gateway verwendet, ein direkter Anschluss des LIN-Netzwerks an den FlexRay-Backbone ist nicht vorgesehen.

5.1.4 Diagnose-Client

Um eine Diagnose bzw. eine Update-Programmierung außerhalb des Fahrzeug-Kommunikationssystems anzustoßen, wird ein Diagnose-Client (Diagnosetester) benötigt. Da der Zugang drahtlos über Bluetooth erfolgt, kommt für diesen Zweck ein Notebook mit einer Bluetooth-Schnittstelle zum Einsatz. Der Diagnose-Client läuft unter einem Debian Linux. Über BlueZ, einem freien und kostenlosen Bluetooth-Stack, wurde die Bluetooth-Schnittstelle eingebunden. Mit Hilfe von BlueZ sind die Entwicklung eigener Bluetooth-Anwendungen unter Linux und der Zugriff auf Bluetooth-spezifische Kommunikationseigenschaften (z.B. Sicherheitsmechanismen oder ein eigenes Diagnoseprofil) möglich.

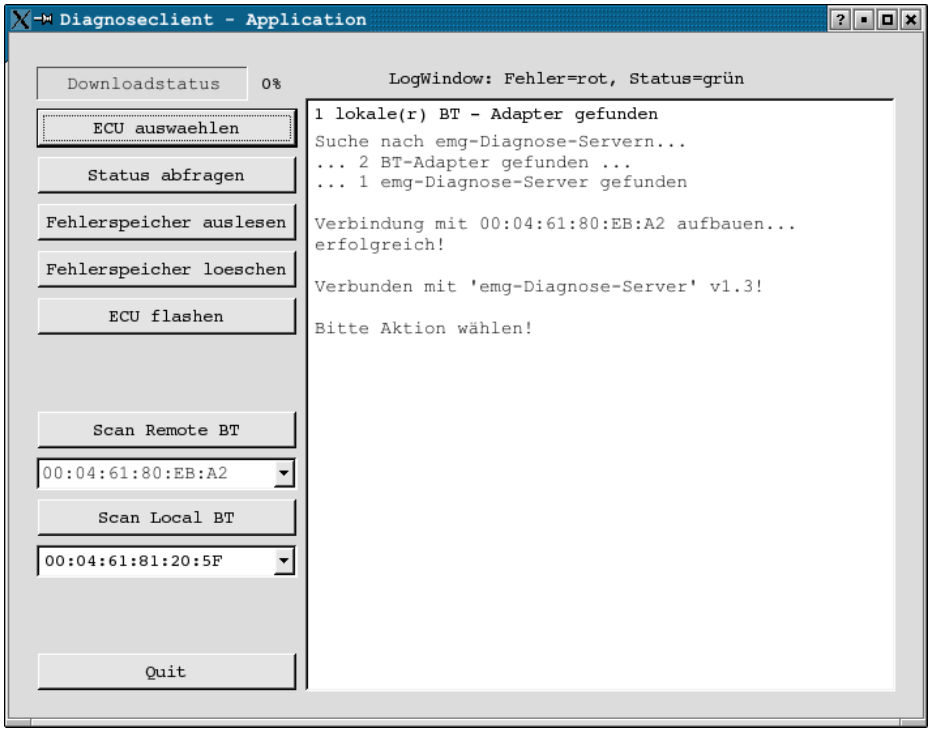

Abbildung 39: Diagnosetester-Applikation unter Qt

Die Diagnoseapplikation wurde als Konsolenanwendung entwickelt, um die Anforderungen an die Hardware möglichst gering zu halten. Darüber hinaus wird mit der Beschränkung auf das zur Diagnosekommunikation Notwendige eine relative Unabhängigkeit zum Linuxsystem erreicht. Gleichzeitig unterstützt dies eine leichte Portierbarkeit auf andere Systeme (z.B. Embedded Systeme oder Windows). Für eine komfortablere Bedienung wurde eine grafische Benutzeroberfläche unter Qt[25] [Qt] entwickelt (siehe Abbildung 39). Qt ist ebenfalls für andere Plattformen verfügbar, so dass die entwickelte Diagnoseapplikation nicht zwangsläufig auf Linux festgelegt ist.

5.2 FlexRay Kommunikationszyklus

Zeitgesteuerte Datenbusse wie FlexRay bieten im Gegensatz zu ereignisgesteuerten Datenbussen den Vorteil der präzisen Vorhersage über die Zeitpunkte des Eintreffens von Nachrichten, Netzlasten und Verzögerungszeiten. Im Gegenzug erfordern sie einen höheren Aufwand beim Entwurf des Kommunikationssystems. Die sendenden Busteilnehmer und der zeitliche Zugriff der einzelnen Teilnehmer auf das Busmedium müssen vorher festgelegt werden. Solch ein Ablauf einer sich wiederholenden Kommunikationsrunde wird als Schedule (dt. Zeitplan) bezeichnet und ist allen Busteilnehmern bekannt. Zum Verständnis des Schedules für den FlexRay-Clusters des Labor-Kommunikationsnetzwerks ist der Aufbau des FlexRay-Protokolls notwendig, der im folgenden Abschnitt erläutert wird. Für den

[25] Qt ist eine Klassenbibliothek und Entwicklungsumgebung für die plattformübergreifende Programmierung graphischer Benutzeroberflächen (GUI) unter C++ und wurde von der Firma Trolltech entwickelt. Qt existiert in verschiedenen Varianten für eine Vielzahl von Betriebssystemen.

zeitgesteuerten Zugriff auf das Busmedium ist es notwendig, dass jeder Busteilnehmer über die gleiche globale Zeitbasis des Bussystems verfügt. Der zeitgesteuerte Datenbus stellt dafür geeignete Mechanismen zur Verfügung. Die Uhrensynchronisation bei FlexRay und das dahinter stehende Konzept werden bei [Temple05] beschrieben.

5.2.1 Aufbau des FlexRay Protokolls

Der Schedule, mit dem bei FlexRay der Zugriff auf das Busmedium und die Datenübertragung organisiert werden, legt fest, welche Signale zu welchem Zeitpunkt mit was für einer Periode innerhalb eines Clusters (dt. Gruppe) übertragen werden und wird als Gruppenzyklus (engl. Cluster Cycle) bezeichnet. Der Cluster Cycle kann wiederum in einzelne Kommunikationszyklen (engl. Communication Cycle) unterteilt werden. Jeder dieser Kommunikationszyklen kann aus einem statischen Segment, einem optionalen dynamischen Segment, einem ebenfalls optionalen Symbol Window und der Network Idle Time (NIT) bestehen. Die einzelnen Segmente setzen sich aus einer festgelegten Anzahl von Abschnitten (engl. Slots) zusammen, die zur Datenübertragung der FlexRay-Frames genutzt werden. Die Slots enthalten eine bestimmte Anzahl von Makroticks, deren Grenzen als Action Points bezeichnet werden. Diese Action Points bestimmen den Beginn und im dynamischen Segment auch das Ende einer Datenübertragung. Die Macroticks setzen sich aus einer festen Anzahl von Mikroticks zusammen, die die kleinste gemeinsame Zeitbasis innerhalb eines FlexRay-Clusters darstellen und deren Länge in den verschiedenen Knoten unterschiedlich sein können. Die Einzelheiten des FlexRay-Protokolls sind in [FlexRay05] beschrieben.

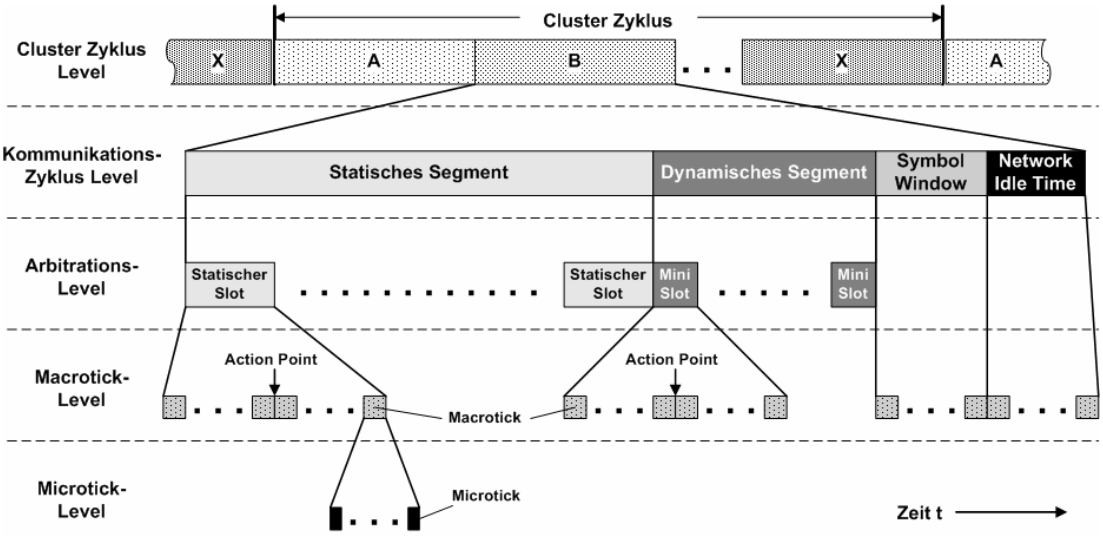

Abbildung 40: Aufbau eines Kommunikationszyklus

Der FlexRay-Datenbus ermöglicht eine ein- oder zweikanalige Datenübertragung. Bei Verwendung von zwei Kanälen verfügt jeder Kanal über seine physikalisch eigene Busleitung. Der zweite Kanal kann je nach Anwendung zur Erhöhung der Datenübertragungsrate oder Erhöhung der Sicherheit durch Redundanz genutzt werden. Bei der Auslegung des Kommunikationszyklus gelten die Vereinbarungen wie Dauer und Länge der einzelnen Segmente für beide Kanäle. Die FlexRay-

Frames, die in einem bestimmten Slot auf den beiden Kanälen gesendet werden, können jedoch unterschiedlich sein. Damit lässt sich die Datenübertragungsrate verdoppeln. Wird in einem Slot die gleiche Nachricht auf beiden Kanälen versendet, so erreicht man eine redundante Übertragung, die sich zur Erhöhung der Ausfallsicherheit nutzen lässt.

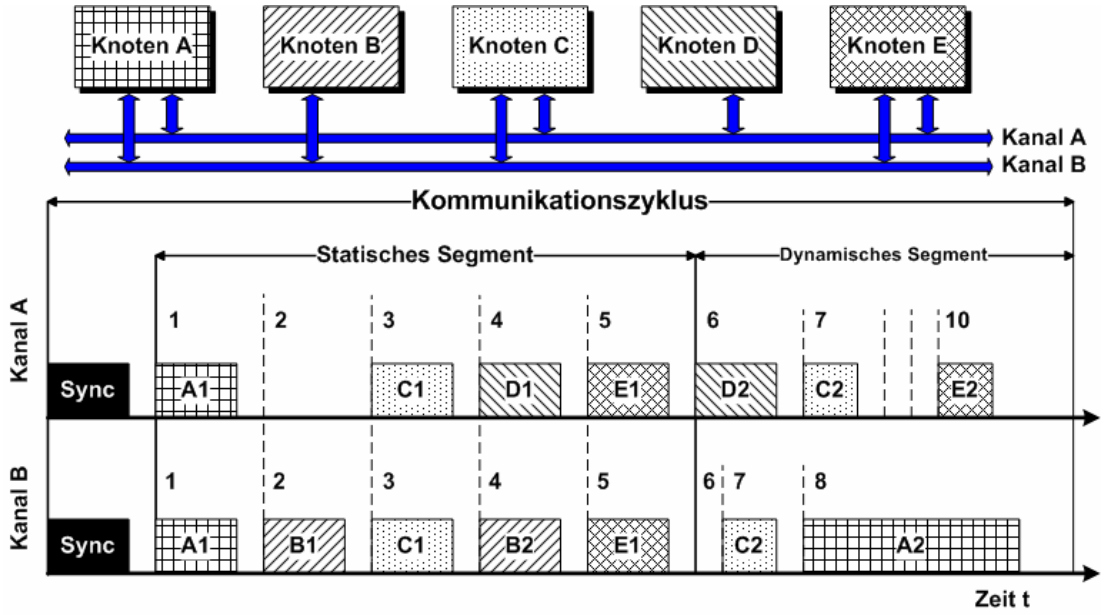

Abbildung 41: Beispiel für einen Kommunikationszyklus-Ablauf

In Abbildung 41 ist ein beispielhafter Ablauf eines Kommunikationszyklus für einen möglichen FlexRay-Cluster, bestehend aus fünf Knoten, dargestellt. Ein Kommunikationszyklus beginnt mit einem Sync-Frame, der für die Synchronisation genutzt wird und die Kommunikation startet. Die Zeitslots werden durch ihre Slot-ID identifiziert, die im statischen und im dynamischen Teil verwendet werden. Das Beispiel in Abbildung 41 zeigt solch eine gemischte Kommunikation mit statischem und dynamischem Segment, bei der nicht jeder Slot zur Datenübertragung genutzt wird. Dazu verdeutlicht das Beispiel gut die unterschiedliche Verwendung der beiden FlexRay-Kanäle und die flexible Nutzung des dynamischen Segments.

5.2.1.1 Statisches Segment

Innerhalb des statischen Segments wird der Zugriff auf den FlexRay-Datenbus über ein Zeitmultiplex-Verfahren geregelt (TDMA). Das statische Segment unterteilt sich in Slots gleicher Länge, deren Anzahl zur Designphase festgelegt wird. Jeder Slot ist durch einen Identifier (ID) gekennzeichnet. In diesem Slot kann genau ein festgelegter Knoten einen Nachrichten-Frame versenden. Die Festlegung, welcher Knoten in welchem Slot sendet, ist im Schedule verankert und wiederholt sich in jedem Cluster Zyklus. Bei zweikanaligem Betrieb werden Nachrichten, die zur Synchronisierung genutzt werden, auf beiden Kanälen verschickt. Andere Nachrichten können auf einem oder auf beiden Kanälen versendet werden (siehe Abbildung 42)

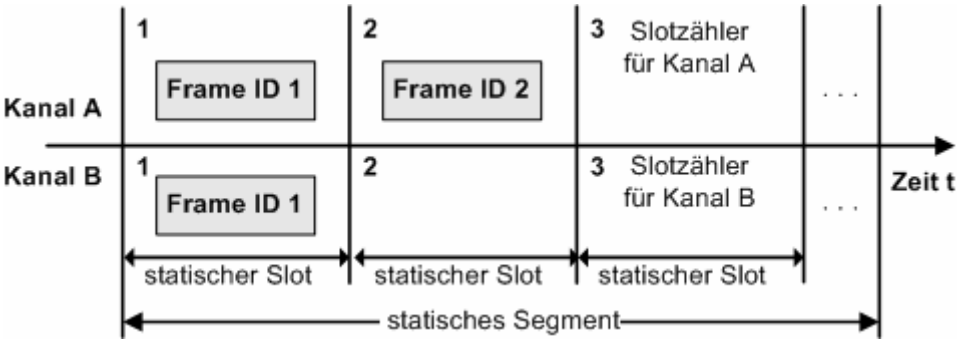

Abbildung 42: Prinzip der TDMA-Runde im statischen Segment

Die Kommunikation im statischen Teil des FlexRay-Kommunikationszyklus ist deterministisch, allerdings auch unflexibel gegenüber Änderungen. Der Kommunikationsablauf wird zur Designphase festgelegt und die Bandbreite entsprechend aufgeteilt. Eine nachträgliche, auch zeitlich beschränkte, Änderung ist zur Laufzeit nicht möglich. Das synchrone Umschalten aller Knoten eines FlexRay-Clusters in einen alternativen Cluster Zyklus wird derzeit vom FlexRay-Protokoll nicht unterstützt. Aus diesem Grund eignet sich das statische Segment mehr zur Übertragung von zyklischen Prozessdaten und weniger zur Übertragung von Flashdaten. Diese fallen temporär und in großen Datenmengen an, so dass eine permanente Reservierung von Bandbreite im statischen Teil sehr ineffektiv wäre.

5.2.1.2 Dynamisches Segment

Im dynamischen Segment wird der Buszugriff über ein Minislotting-Verfahren geregelt, bei dem die Zuteilung des Busmediums prioritätengesteuert geschieht. Dazu besitzt jeder Frame, der im dynamischen Segment gesendet werden soll, eine eindeutige ID, die mit der Minislot-ID korrespondiert. Ist der betreffende Minislot aktiv, so kann die Nachricht gesendet werden. Ist ein Versenden der Nachricht im aktuellen Kommunikationszyklus nicht notwendig, so wird der Minislot-Zähler nach einer gewissen Zeitspanne inkrementiert und der nächste Minislot ist aktiv. Den Vorgang des Minislotting-Verfahrens zeigt Abbildung 43.

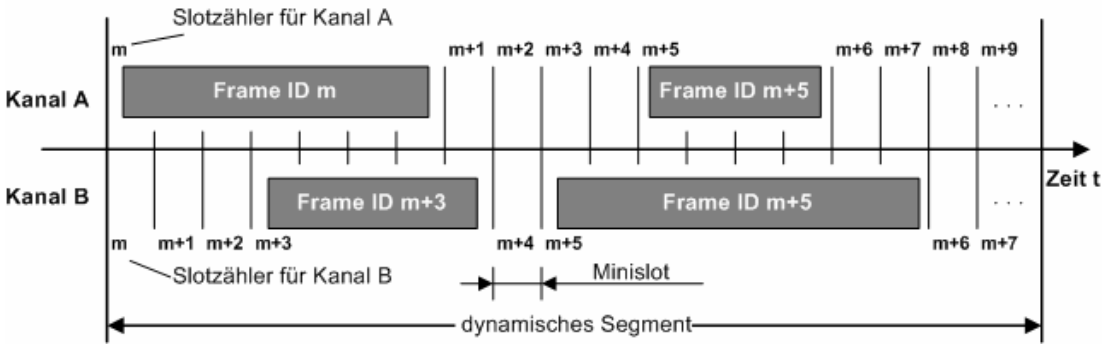

Abbildung 43: Prinzip des Minislotting-Verfahren im dynamischen Segment

Mit Hilfe dieses Mechanismus kann die zur Verfügung stehende Bandbreite im dynamischen Teil wesentlich effektiver als im statischen Teil genutzt werden. Außerdem darf im dynamischen Teil die Nachrichtenlänge auf beiden Kanälen und

in den Kommunikationszyklen unterschiedlich sein. Allerdings geht diese Flexibilität auf Kosten des Determinismus. Werden in mehreren Minislots Nachrichten versendet, so ist es möglich, dass der dynamische Teil beendet ist, bevor der Minislot-Zähler seinen maximalen Wert erreicht. Noch nicht gesendete Nachrichten müssen dann in einem späteren Kommunikationszyklus versendet werden.

Der dynamische Teil bietet sich zur Übertragung von Diagnose- und Flashdaten an. Es muss für die Diagnose- und Flashkommunikation keinerlei Bandbreite reserviert werden, sondern nach Bedarf steht bei geeigneter Priorisierung die gesamte Bandbreite des dynamischen Segments zur Verfügung. Die Einschränkung des dynamischen Teils durch die fehlende Eigenschaft des Determinismus ist für diese Art der Kommunikation nicht von Bedeutung.

5.2.1.3 Symbol Window und Network Idle Time

Das FlexRay-Protokoll berücksichtigt die Möglichkeit des einfachen Signalisierens von bestimmten Ereignissen. Im optionalen Symbol Window können über den FlexRay einige definierte Symbole wie das Collision Avoidance Symbol (CAS), das Media Access Test Symbol (MTS) oder das Wake Up Symbol (WUS) gesendet werden. Eine Arbitrierung im Symbol Window ist durch FlexRay nicht vorgesehen. Beim Fall, dass mehrere Knoten das Symbol Window nutzen möchten, muss durch höhere Protokollschichten verhindert werden, dass Teilnehmer gleichzeitig senden. Im Rahmen dieser Arbeit wurde auf das Symbol Window verzichtet und nicht weiter betrachtet.

Die Network Idle Time stellt immer das Ende eines Kommunikationszyklus dar. Während dieser Zeit findet keinerlei Kommunikation auf dem Busmedium statt. Die Knoten führen in dieser Zeit die Uhrensynchronisation und Berechnungen zur Zeitkorrektur durch.

5.2.1.4 FlexRay Frame Format

Die Nachrichten (engl. Frames), die innerhalb der einzelnen Slots im statischen und dynamischen Segment versendet werden können, haben alle den in Abbildung 44 dargestellten prinzipiellen Aufbau.

Eine FlexRay-Nachricht beginnt mit einem 5 Bytes langen Header, dessen Struktur in Tabelle 6 beschrieben ist. Dem Header folgt der Abschnitt mit den zu übertragenden Nutzdaten, der bis zu 254 Byte lang sein darf. Auf Grund der Längenkodierung in Datenworten (entspricht 2 Bytes) im Header, besteht der Nutzdatenabschnitt immer aus einer geraden Anzahl von Bytes. Daran schließt sich ein Trailer von 3 Byte Länge an. Er beschließt den FlexRay Frame und enthält einen CRC-Code, mit dem der Frame gegen Bitfehler abgesichert wird.

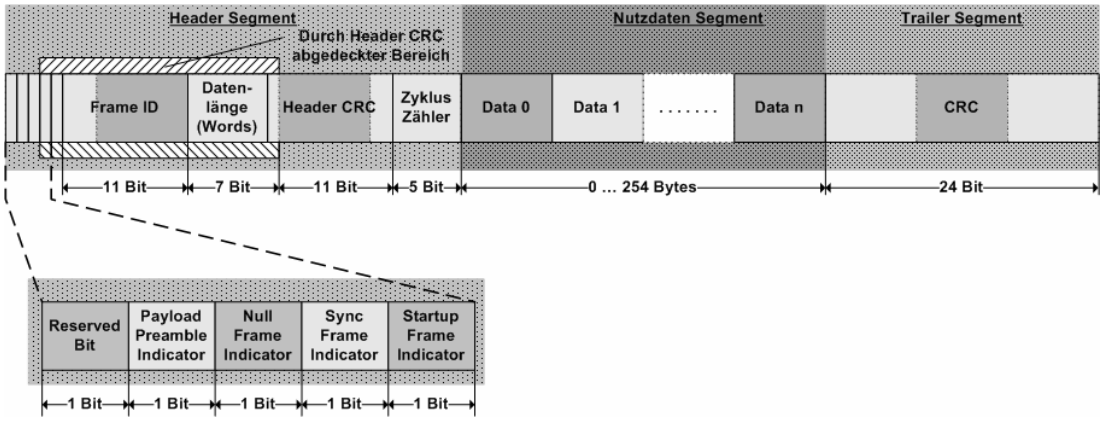

Abbildung 44: FlexRay Frame Format

Bezeichnung	Größe	Beschreibung
Reserved Bit	1 Bit	Für zukünftige Protokollerweiterungen (derzeit auf „0“)
Payload Preamble Indicator	1 Bit	Signalisiert, ob die Nutzdaten einen optionalen Vector enthalten. • Statischer Teil – Netzwerkmanagement Vector (NMVector) • Dynamischer Teil – Message ID Vector
Null Frame Indicator	1 Bit	Signalisiert, ob das Nutzdatensegment gültige Daten enthält.
Sync Frame Indicator	1 Bit	Signalisiert, ob das Frame zur Synchronisierung der Cluster Kommunikation genutzt wird.
Startup Frame Indicator	1 Bit	Signalisiert, ob das Frame ein Startframe ist. Diese haben eine spezielle Rolle innerhalb des Startmechanismus.
Frame ID	11 Bit	Definiert die Slotposition im statischen und die Priorität im dynamischen Segment.
Payload Length	7 Bit	Definiert die Größe der Nutzdaten als Word (2 Byte)
Header CRC	11 Bit	Enthält einen CRC, der über Sync Frame Indicator, Startup Frame Indicator, Frame ID und Payload Length gebildet wird.
Cycle Count	6 Bit	Enthält den Zykluszähler.

Tabelle 6: Aufbau des FlexRay Frame Headers

Eine Besonderheit beinhaltet der Nutzdatenabschnitt, wenn im Header das Flag „*Payload Preamble Indicator*“ gesetzt ist. Abhängig vom jeweiligen Segment bedeutet dies das Vorhandensein eines Network Management Vectors (NMVector) oder einer Message ID. Innerhalb des statischen Segments kann ein FlexRay-Frame bei gesetztem Flag im Header einen NMVector enthalten. Seine Länge ist als Parameter zwischen 0 und 12 Bytes festgelegt und gilt für alle Knoten eines Clusters. Mit Hilfe des NMVector können netzwerkweite Ereignisse wie beispielsweise das „Aufwachen“ oder das „Schlafenlegen“ des Busses signalisiert werden.

Dagegen bedeutet das gesetzte Header-Flag „*Payload Preamble Indicator*“ im dynamischen Segment, dass die ersten beiden Bytes der Nutzdaten eine Message ID enthalten. Diese Message ID kann von der Applikation zur Identifizierung und damit Filterung der nachfolgenden Daten aus dem Nutzdatensegment verwendet werden.

Die Abbildung 45 verdeutlicht die Zusammenhänge des gesetzten Header-Flags „*Payload Preamble Indicator*“ für das statische und das dynamische Segment.

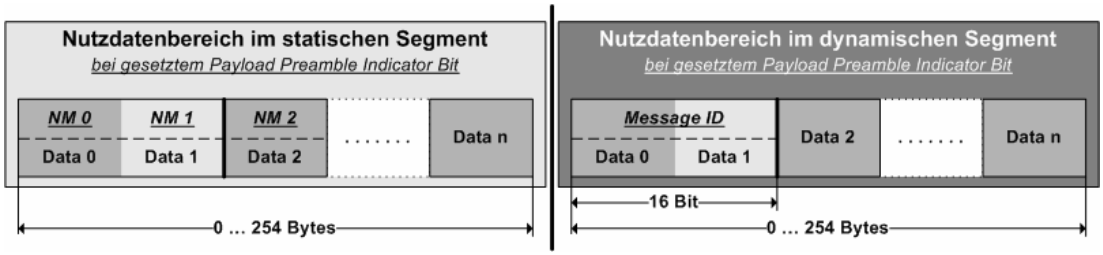

Abbildung 45: Optionale Nutzung eines NMVectors oder einer MessageID im Nutzdatenbereich

5.2.2 Simulieren der Standardkommunikation

Der FlexRay spielt als Backbone innerhalb der Architektur des Kommunikationssystems eine zentrale Rolle. Die Hauptaufgabe des Backbones besteht im Austausch der Daten zwischen den verschiedenen Bussystemen. Durch die Dominanz des CAN-Datenbusses in heutigen Kraftfahrzeugen ist davon auszugehen, dass neben den Daten von FlexRay-Steuergeräten überwiegend CAN-Daten über den FlexRay-Backbone transportiert werden. Da es sich bei einem großen Teil der Daten, die über den CAN-Bus gesendet werden, um zyklische Prozessdaten handelt, wird das Tunneln dieser Kommunikation über den FlexRay im statischen Segment vorgenommen. Diese zyklische Kommunikation wird hier als Standardkommunikation bezeichnet, während eine Diagnose-Kommunikation eine sporadische Art der Kommunikation darstellt.

5.2.2.1 CAN-Kommunikation über den FlexRay-Bus

Die Möglichkeiten zum Tunneln einer CAN-Kommunikation über das statische Segment des FlexRay werden im Wesentlichen durch die FlexRay-Framelänge bestimmt, die im statischen Segment clusterweit vereinbart wurde und für alle statischen Slots identisch ist. Da sich FlexRay zum Zeitpunkt der Arbeit noch im Prototypenstadium befindet, liegen noch keine Erfahrungen seitens der Automobilhersteller für eine optimale Framelänge eines FlexRay-Backbones vor. Es ist davon auszugehen, dass die statischen FlexRay-Frames mehr als die CAN-üblichen 8 Bytes nutzen werden, um den Datendurchsatz zu erhöhen. Allerdings würde wiederum eine maximale Ausnutzung der FlexRay-Frames mit 254 Bytes die mögliche Anzahl verfügbarer Slots in einer akzeptablen Zykluszeit stark einschränken. Daher ist eine Länge der statischen Frames im unteren Bereich der möglichen Framelänge zu erwarten. Erste Implementierungen seitens der Fahrzeughersteller bei Prototypenfahrzeugen nutzen eine Framelänge von 24 Bytes im statischen Segment.

Bei den FlexRay-Controllern des als Backbone eingesetzten FlexRay-Clusters handelt es sich um Prototypen. Sie sind als FPGA mit eingeschränktem Funktionsumfang realisiert. Die Datenpuffer, die zum Datenaustausch mit dem FlexRay-Bus genutzt werden, verfügen statt der benötigen Größe von 254 Bytes nur über eine Größe von 32 Bytes. Die maximale Größe der FlexRay-Frames, die im eingesetzten Cluster verwendet werden darf, ist somit auf 32 Bytes festgelegt.

Allerdings stellen die 32 Bytes einen sehr guten Kompromiss zwischen Durchsatz und möglicher Slotanzahl dar, so dass dies keine große Einschränkung bedeutet.

Die verfügbaren 32 Bytes lassen sich theoretisch sehr gut auf vier CAN-Nachrichten á 8 Bytes aufteilen. Allerdings ist dabei der zeitgesteuerte zyklische Ablauf der Kommunikation beim statischen FlexRay-Segment zu berücksichtigen. Im Gegensatz dazu kann beim ereignisgesteuerten CAN-Bus nicht vorausgesetzt werden, dass Daten, die über den FlexRay getunnelt werden sollen, zu jedem Zeitpunkt des Zyklus verfügbar sind. Daher sind zusätzliche Informationen notwendig, anhand derer die Applikation entscheiden kann, ob die CAN-Nachrichten im FlexFray-Frame gültig sind. Im Rahmen dieser Arbeit wurden dafür zwei verschiedene Herangehensweisen betrachtet, hier als asynchron und isochron bezeichnet, die im Folgenden vorgestellt werden.

5.2.2.2 Asynchrones Tunneln von CAN-Nachrichten

Bei der asynchronen Übertragung von CAN-Nachrichten über den FlexRay-Datenbus werden ein oder mehrere FlexRay-Frames als Container reserviert, die verschiedene vorhandene CAN-Nachrichten eines CAN-Datenbusses aufnehmen. Eine Nachricht des Standard-CAN-Datenbusses belegt innerhalb des Datencontainers 10 Byte und enthält neben den 8 Byte Daten zwei weitere Bytes, die zur Identifikation benötigt werden (siehe Abbildung 46). In den zwei Identifikationsbytes (16 Bit) lassen sich die 11 Bit CAN-ID und mit 3 Bit die Datenlänge der CAN-Nachricht (0…8) verschlüsseln. Mit den verbleibenden zwei Bits können drei verschiedene Zustände kodiert werden. Diese lassen sich zur Signalisierung der Gültigkeit der Nachricht oder zur Kennzeichnung des CAN-Datenbusses, von dem die CAN-Nachrichten stammen, verwenden.

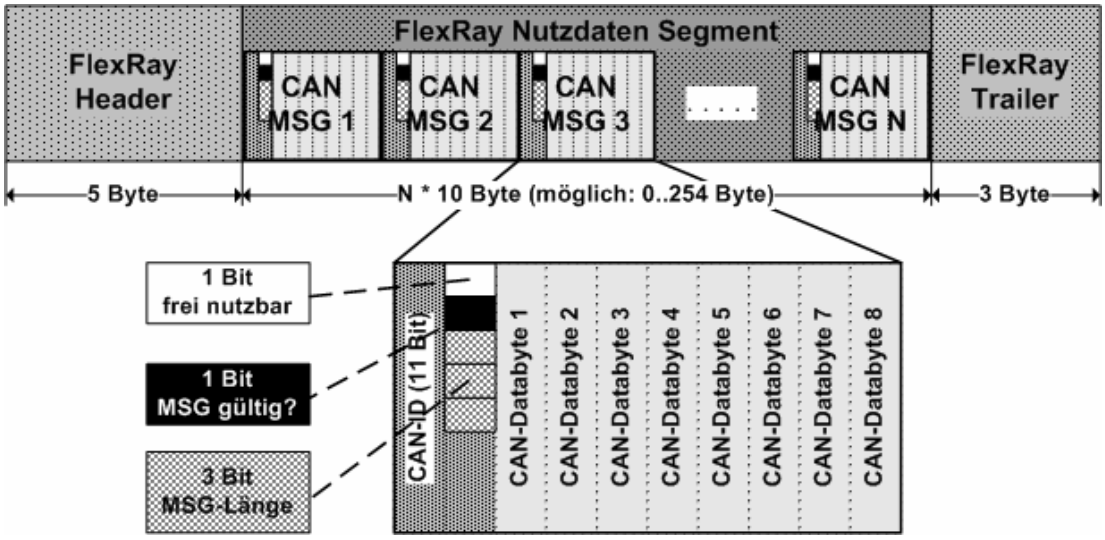

Abbildung 46: Asynchroner CAN-Tunnel über FlexRay

Der Vorteil dieser Art der CAN-Übertragung über den FlexRay-Backbone liegt in der Unabhängigkeit vom Eintreffen der zu transportierenden CAN-Nachrichten. Die

CAN-Nachrichten werden in einem FIFO-Speicher[26] gesammelt, der zu jedem Kommunikationszyklus des FlexRay geleert wird. Durch geeignete Wahl der Zykluszeit und der Speichergröße muss ein Überlaufen des Speichers auch bei maximaler Anzahl zu übertragender CAN-Nachrichten ausgeschlossen sein. Da jede CAN-Nachricht über ihre ID sowie gegebenenfalls über ihren Herkunftsdatenbus eindeutig zuzuordnen ist, existieren keine Beschränkungen auf die Übertragung nur bestimmter, vorher festgelegter, CAN-Nachrichten. Nachteile der asynchronen Übertragung liegen dagegen im erhöhten Overhead der Nachrichten sowie der etwas aufwendigeren Datenverarbeitung im Vergleich zur isochronen Übertragung.

5.2.2.3 Isochrones Tunneln von CAN-Nachrichten

Werden über den FlexRay-Backbone vorwiegend Prozessdaten von einem CAN-Datenbus zu einem anderen CAN-Datenbus transportiert, bietet sich eine isochrone Übertragung an. Prozessdaten werden zyklisch gesendet und sind durch ihre festgelegte äußere Form (CAN-ID, Datenlänge) gekennzeichnet. Bei der isochronen Übertragung bekommen die Prozessdaten einen festen Platz innerhalb des FlexRay-Frame zugewiesen (siehe Abbildung 47).

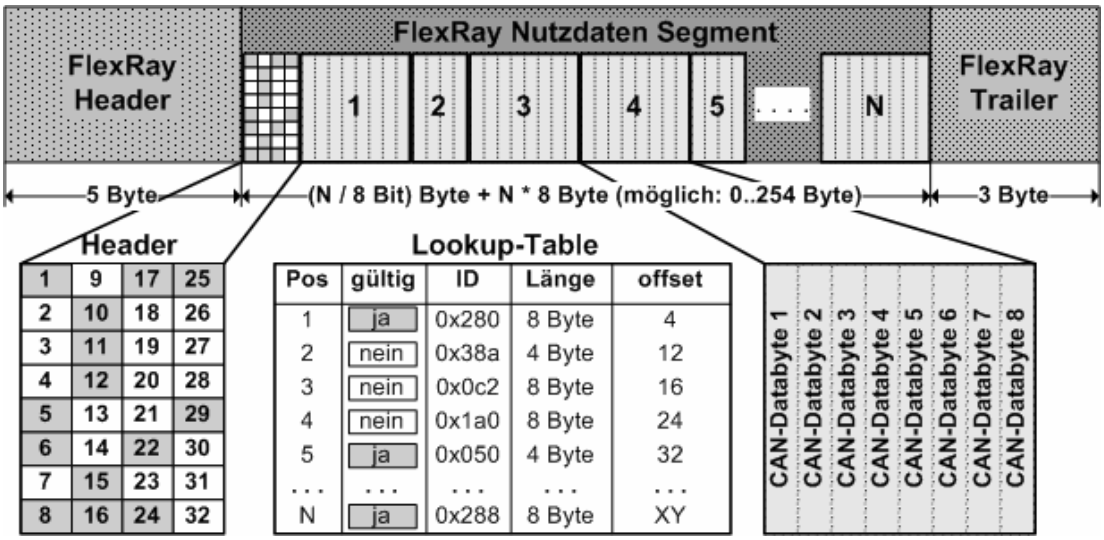

Abbildung 47: Isochroner CAN-Tunnel über FlexRay

Mit diesem Offset innerhalb des Datencontainers sind die CAN-ID und die CAN-Datenlänge fest verknüpft. Über einen Header vor dem Datencontainer wird die Gültigkeit der einzelnen CAN-Nachrichten bitweise kodiert. Dieses Vorgehen ist ähnlich dem Prinzip der Message-Objekte, wie es bei FullCAN-Controllern angewendet wird. Die Message-Objekte sind spezielle reservierte Register im CAN-Controller. Sie werden für eine bestimmte CAN-Nachricht mit CAN-ID und CAN-Datenlänge konfiguriert und sind nur für diese eine Nachricht mit der CAN-ID zuständig. Der Vorteil liegt bei diesem Prinzip n der Signalisierung eintreffender Nachrichten und dem einfachen Zugriff auf den Dateninhalt über die Register. In Verbindung eines FullCAN-Controllers mit dem isochronen Tunneln beschränkt sich die Verarbeitung der Applikation auf eine kurze Auswertung der Signalisierung und

[26] FIFO – First In / First Out - Speicher

einfaches Umkopieren des Dateninhalts an die vorgesehene Stelle innerhalb des FlexRay-Frames.

Durch diese Art der Datenübertragung wird der Overhead des Datencontainers auf den Header begrenzt und ist mit einem Bit pro CAN-Nachricht fast vernachlässigbar. Allerdings gilt dies für jeden Kommunikationszyklus, unabhängig davon, ob CAN-Daten übertragen werden müssen oder nicht. Geht man von einer maximalen Framelänge von 254 Byte aus, können bis zu 31 CAN-Nachrichten mit der maximalen CAN-Datenlänge von 8 Byte im Container transportiert werden. Von den verbleibenden 6 Bytes werden 4 Byte für den Header verwendet, mit dem sich Gültigkeit von 32 CAN-Nachrichten kodieren lassen. Ein weiterer Vorteil neben dem geringen Overhead liegt in dem einfachen Zugriff auf die Daten innerhalb des FlexRay-Frames. Hier reicht ein Maskenvergleich auf ein Bit, um gültige Daten zu identifizieren. Danach genügt das Kopieren der Daten ab einem bestimmten Offset mit einer bekannten Länge, um die Daten zu erhalten.

5.2.2.4 Tunneln einer CAN-Antriebsstrangkommunikation über FlexRay

Bei dem FlexRay-Backbone kann beim derzeitigen Stand der Technik nicht davon ausgegangen werden, dass die gesamte vorhandene Bandbreite für die Diagnosekommunikation zur Verfügung steht. Der Grund liegt in der aufwendigen Umschaltung zwischen verschiedenen Clusterzyklen, die vom FlexRay-Protokoll derzeit nicht vorgesehen ist. Die Diagnosekommunikation wird vielmehr parallel zur Standardkommunikation ablaufen. Daher musste für das Labornetzwerk solch eine Standardkommunikation, wie sie in Fahrzeugen üblich ist, simuliert werden. Zu diesem Zweck stand ein VW Polo (Baureihe PQ 24) mit Vollaustattung[27] zur Verfügung (siehe Abbildung 48). Mit Hilfe eines Canalyzers[28] wurde die CAN-Kommunikation vom Antriebsstrang des VW Polo während verschiedener Messfahrten mitgeloggt. Die Buslast des 500 kBit/s schnellen Antriebs-CAN betrug dabei im Mittel 25%.

Abbildung 48: Versuchsfahrzeug VW Polo (PQ 24)

[27] Vollaustattung bedeutet das Vorhandensein aller für das Modell verfügbaren elektronischen Steuergeräte.

[28] Canalyzer bezeichnet ein Werkzeug der Firma Vector [Vector] zum Überwachen des CAN-Datenbusses. Es erlaubt u.a. das Mithören, Loggen und nachträgliche Wiederabspielen des CAN-Datenverkehrs.

Die Kommunikation in der Domäne „Antriebsstrang“ ist bei der im Labornetzwerk realisierten Backbone-Architektur in sich abgeschlossen. Allein das Gateway-Steuergerät zum Backbone bestimmt, welche Daten diese Domäne verlassen und welche hinein übertragen werden. Diese Eigenschaft kann zur Erhöhung der Sicherheit und zur Entlastung des übrigen Kommunikationssystems genutzt werden. Als Vereinfachung der Annahmen und Simulation eines Worst Case-Szenario wird innerhalb der Untersuchung die vollständige Antriebsstrang-Kommunikation über den FlexRay-Backbone übertragen. In einem realen Fahrzeug könnte damit die Daten der Antriebsstrang-Steuergeräte beispielsweise dem Kombi-Steuergerät, der Domäne „Infotainment“ oder dem Diagnosesteuergerät verfügbar gemacht werden.

Durch die Übertragung der vollständigen Kommunikation wird der FlexRay-Backbone stärker belastet. Dies erhöht die Aussagekraft der Untersuchung, da ein Backbone nicht nur eine, sondern mehrere Domänen bedienen muss. Um dem weiter Rechnung zu tragen, wird die Simulation der Standardkommunikation über den Backbone mit der zweifachen Antriebsstrang-Kommunikation realisiert. Unter Berücksichtigung des Prototypen-Stadiums des eingesetzten FlexRay-Clusters, dem mit 5 MBit/s nur die Hälfte der spezifizierten Bandbreite zur Verfügung steht, sind das realistische Annahmen hinsichtlich zukünftiger Anforderungen an einen Backbone.

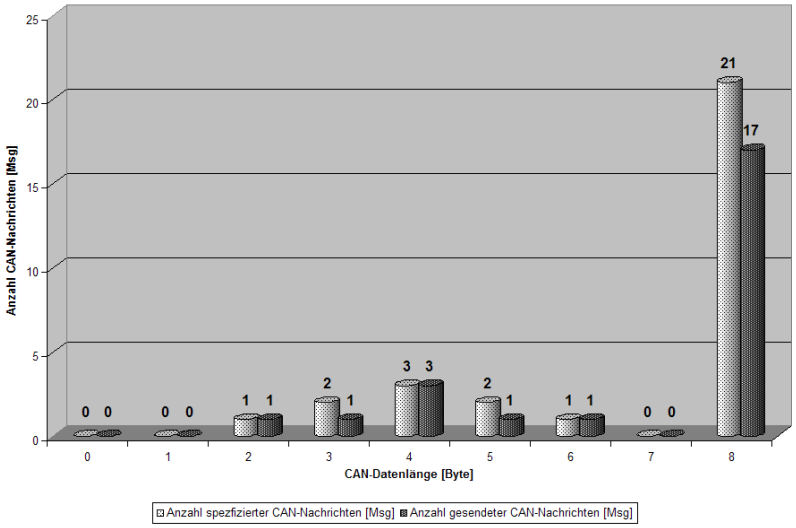

Abbildung 49: Auswertung der PQ 24 Antriebsstrangkommunikation

Die Simulation der Standardkommunikation wurde als isochroner CAN-Tunnel realisiert. Die Auswertung der CAN-Datenbasis für den PQ 24 ergab 30 verschiedene CAN-Nachrichten auf dem Antriebsstrang mit Datenlängen zwischen zwei und acht Byte. Die ungefähren Zykluszeiten der Nachrichten auf dem CAN-Bus liegen zwischen zehn Millisekunden und einer Sekunde. Die CAN-Mitschnitte vom Antriebsstrang des Versuchsfahrzeugs zeigen, dass dort 24 der spezifizierten 30 CAN-Nachrichten gesendet werden (siehe Abbildung 49). Die Ursache für die „fehlenden“ Nachrichten liegt in den unterschiedlichen Ausstattungsvarianten der einzelnen Modelle. Beispielsweise besitzt das Versuchsfahrzeug eine manuelle Schaltung anstelle eines Automatikgetriebes, so dass dem Fahrzeug dieses Steuergerät fehlt. Zur Auslegung des CAN-Tunnels über den FlexRay wurde

allerdings von den 30 spezifizierten CAN-Nachrichten und der minimalen Zykluszeit von 10 Millisekunden ausgegangen.

Die maximale Größe der statischen FlexRay-Frames ist auf 32 Byte festgelegt, was für das isochrone Tunneln der CAN-Antriebsdaten ausreichend ist. Dabei wird ein Byte für den Header benötigt, so dass noch 31 Byte pro Frame für die zu übertragenden CAN-Nachrichten zur Verfügung stehen. Diese wurden auf je drei CAN-Nachrichten á 8 Byte aufgeteilt. Die verbleibenden 7 Byte der einzelnen Frames werden optimal mit CAN-Nachrichten gefüllt, die weniger als 8 Byte Länge besitzen. Für eine Übertragung der vollständig spezifizierten Antriebsstrang-kommunikation des PQ 24 sind damit sieben statische FlexRay-Frames mit einer Länge von 32 Byte notwendig.

Im Rahmen dieser Arbeit beschränkt sich der Clusterzyklus auf genau einen Kommunikationszyklus mit einer Zykluszeit von 10 Millisekunden. Der Einsatz mehrerer Kommunikationszyklen innerhalb eines Clusterzyklus bietet weiteres Optimierungspotenzial hinsichtlich der Zykluszeiten einzelner CAN-Nachrichten, was die Bandbreite auf dem FlexRay-Backbone effektiver nutzt.

Zur Simulation der Standardkommunikation auf dem FlexRay-Backbone wird je ein aufgezeichneter CAN-Mitschnitt vom Antriebsstrang mittels eines Canalyzers in einer Endlosschleife auf zwei CAN-Datenbusse eingespeist. Jeder dieser beiden CAN-Datenbusse ist an einen der ARM-Knoten angeschlossen, der die Antriebs-strang-Kommunikation empfängt und über den FlexRay sendet. Diese CAN-Nachrichten werden vom jeweils anderen ARM-Knoten vom FlexRay gelesen und über seinen zweiten CAN-Kanal wieder ausgegeben. Mit Hilfe des Canalyzers kann dabei kontrolliert werden, ob CAN-Nachrichten verloren gehen.

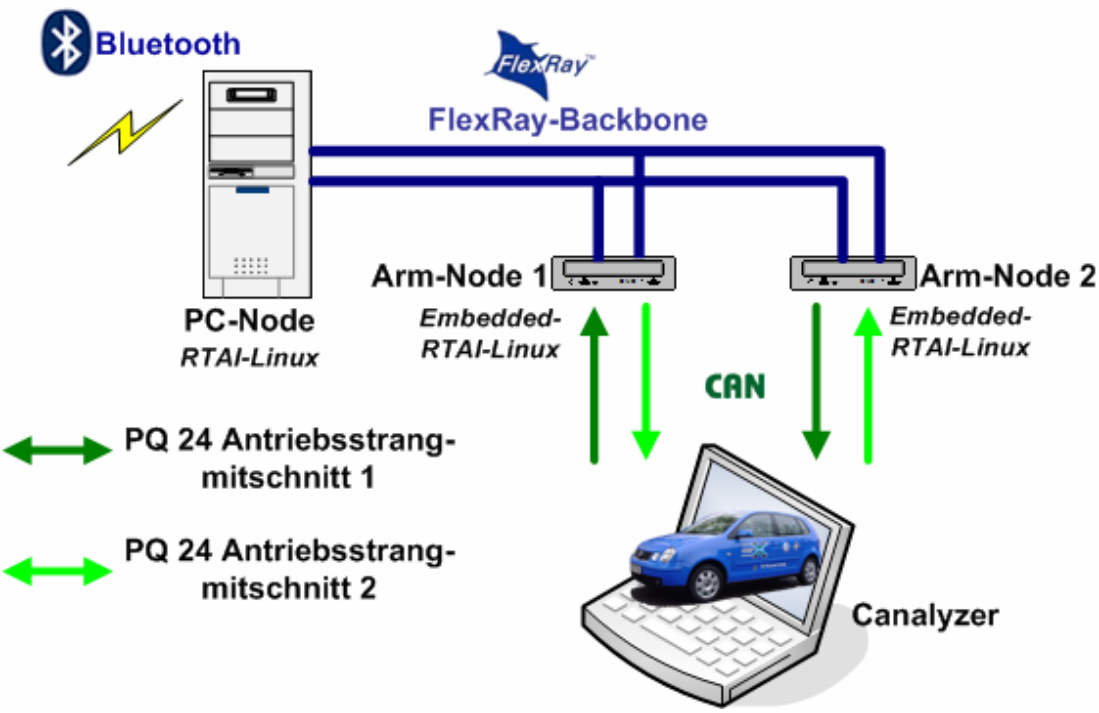

Abbildung 50: Aufbau zur Simulation der Standardkommunikation

In Abbildung 50 ist die Realisierung der Standardkommunikation mit dem eingesetzten FlexRay-Cluster gezeigt. Durch die Taktung des Prototypen-FlexRay von 5 MHz benötigt eine Bitzeit auf dem Bus 0,2 µs. Zusammen mit der FlexRay-Framelänge von 32 Byte (brutto: 40 Byte = 320 Bit) resultiert eine statische Slotlänge von 90 µs[29]. Die Kommunikationszykluszeit orientiert sich an der minimalen Zykluszeit der CAN-Prozessdaten und ist auf 10 Millisekunden festgelegt. Für den statischen Teil des Kommunikationszyklus eines Backbones sollen mindestens 60 % der Bandbreite verwendet werden, um den Vorteil des Determinismus ausnutzen zu können. Der Kommunikationszyklus des eingesetzten FlexRay-Clusters im Labornetzwerk verfügt mit dieser Vorgabe über 70 statische Slots. Davon werden 14 dieser Slots zur Übertragung der zweifachen Kommunikation vom Antriebsstrang verwendet. Damit bleiben 56 Slots á 32 Byte Nutzdatenlänge für weitere Kommunikationsaufgaben verfügbar, was genügend Reserven bietet. Der statische Teil nimmt mit diesen Festlegungen 6,3 Millisekunden des gesamten Kommunikationszyklus von 10 Millisekunden ein. Damit verbleiben dem dynamischen Teil etwas mehr als 3,5 Millisekunden pro Kommunikationszyklus und damit fast 40 % der verfügbaren Bandbreite. Diese wird im Rahmen dieser Arbeit für die Diagnosekommunikation und zur Update-Programmierung verwendet.

5.2.3 Realisierung der Diagnose- und Flashkommunikation

Eine externe Diagnosekommunikation wird im Vergleich zur ständigen internen Prozessdatenkommunikation in einem Kraftfahrzeug relativ selten benötigt. Daher ist es sehr ineffektiv für den Austausch von Diagnosenachrichten ständig Bandbreite auf dem FlexRay-Backbone zu reservieren und vorzuhalten. Zu dem wären damit die Diagnosenachrichten auf die feste Framelänge des statischen Segments festgelegt, die relativ klein im Vergleich zur maximal möglichen Länge sein wird. Die Folgen wären eine starke Erhöhung des Overheads während einer Update-Programmierung und die Verlängerung der Dauer eines Flashvorgangs.

Aus diesen Gründen bietet sich für die Diagnosekommunikation die hohe Flexibilität des dynamischen Teils des Kommunikationszyklus zur Nutzung an. Für die Realisierung dieser Kommunikation im dynamischen Teil wurden einige Vorüberlegungen und Untersuchungen angestellt. Diese ergaben, dass die gewöhnliche Diagnosekommunikation relativ geringe Ansprüche an einen Kommunikationskanal bezüglich der Bandbreite stellt und als fast symmetrisch gelten kann. Symmetrisch bedeutet, dass das Verhältnis der in das Fahrzeug gesendeten Daten ausgeglichen zu den vom Fahrzeug empfangenen Daten ist. Beispiele für eine solche Art der Diagnosekommunikation sind das Auslesen des Fehlerspeichers oder einzelne Kommandos wie „Löschen des Fehlerspeichers“ und die dazugehörigen Quittierungen.

Demgegenüber steht als Spezialfall der Diagnosekommunikation die Übertragung von Software zur Programmierung der Steuergeräte im Kraftfahrzeug. Diese Art der Kommunikation ist im höchsten Maße asymmetrisch. Die zu programmierenden Daten, die in das Fahrzeug übertragen werden, haben bei heutigen Kraftfahrzeugen

[29] Auch diese relative lange Zeit für einen statischen Slot resultiert aus dem Prototypenstadium des FlexRay-Clusters. Spätere Versionen werden erhebliche kürze Slotlängen für die 32 Byte Nutzdaten benötigen.

pro Steuergerät eine Größe von mehreren Kilobytes bis zu einigen Megabytes. Diese Mengen an Daten müssen sehr schnell und sicher in das Fahrzeug gelangen. Daher bietet sich in dem Fall die volle Ausnutzung der maximalen Framelänge von 254 Bytes bei FlexRay an.

Der gesamte dynamische Teil eines Kommunikationszyklus des FlexRay-Backbones wird im Rahmen dieser Arbeit zur Diagnosekommunikation verwendet. Andere Applikationen, die eventuell auch den dynamischen Teil nutzen, konnten innerhalb der Vorüberlegungen nicht identifiziert werden. Über eine geeignete Priorisierung würden sich aber auch diese Applikationen in einem Kommunikationszyklus parallel zur Diagnose betreiben lassen. Allerdings sollten Anwendungen, die parallel zur Update-Programmierung ablaufen und kommunizieren, prinzipiell aus Sicherheitsgründen ausgeschlossen werden. Zum einen verringern sie die zur Verfügung stehende Bandbreite und erhöhen die Programmierzeit. Zum anderen befinden sich die gerade zu programmierenden Steuergeräte während der Programmierung in einem kritischen Zustand, der die gesamte Domäne betreffen kann. Daher sollten während der Programmierung vom Abruf weiterer Funktionen und zusätzlicher Kommunikation abgesehen werden.

5.2.3.1 Identifizierung der Diagnosedaten

Für eine geeignete Priorisierung der Diagnosedaten innerhalb des dynamischen Segments vom FlexRay-Backbone ist eine Klassifizierung und Wertung erforderlich. Dabei wird im Folgenden der Off-Board-Diagnosetester, der eine Diagnose initiiert, als Diagnose-Client bezeichnet. Dieser muss sich mit dem Fahrzeug, genauer mit einem speziellen On-Board-Steuergerät verbinden, welches die Schnittstelle zwischen Fahrzeug und Umwelt bildet und als Diagnose-Server bezeichnet wird. Der Diagnose-Client muss sich gegenüber dem Diagnose-Server authentifizieren und steuert anschließend den weiteren Ablauf der Kommunikation. On-Board läuft die Diagnosekommunikation zwischen dem Diagnose-Server und einem Steuergerät ab. Diese On-Board-Diagnosekommunikation wird dabei in drei Arten unterteilt, die in Tabelle 7 aufgelistet sind. Die Off-Board-Diagnosekommunikation zwischen Diagnose-Client und Diagnose-Server wurde im Rahmen der Arbeit mittels einer klassischen Client/Server-Architektur realisiert. Die Art der zu übertragenden Daten spielen bei der Off-Board-Kommunikation eine untergeordnete Rolle.

Diagnoseart	Diagnose-Server	Steuergerät	Ca. Größe [Byte]	Priorität
Kommando (Com)	X	–	1 – 2	Sehr hoch
Quittung (Ack)	X	X	1 – 2 evtl. Fehlercode: + 2	Hoch
Daten (Data)	X	X	Beliebig	Mittel

Tabelle 7: Arten von Diagnosedaten bei der On-Board-Diagnosekommunikation

5.2.3.2 Umsetzung der Diagnosekommunikation

Auf Grundlage der Klassifizierung aus Tabelle 7 ergeben sich die Priorisierungen über die Message-ID für das dynamische Segment des Kommunikationszyklus und die Framelängen der einzelnen Nachrichtenarten. Die Diagnosenachrichten, die Kommandos vom Diagnose-Server enthalten, werden als Nachrichten vereinbart, die die höchste Priorität innerhalb des dynamischen Segments besitzen. Danach folgen die Nachrichten der Steuergeräte, die für Quittierungen genutzt werden können. Die Anzahl dieser Nachrichten muss der Anzahl der angeschlossenen Steuergeräte am Backbone entsprechen. Als Nachricht mit nächst höherer Priorität folgt die Quittierungsnachricht, die vom Server genutzt werden kann. Alle diese Nachrichten, die zu Quittierungen und Kommandos genutzt werden, benötigen eine relativ geringe Anzahl von Bytes. Allerdings würde eine Framelänge von einem oder zwei Bytes die Effizienz der Übertragungsrate einschränken bei nur einer kleinen Verkürzung der Latenzzeiten. Aus diesem Grund wurden diese Nachrichtentypen auf eine Framelänge von 8 Byte festgelegt. Damit bleiben Reserven für die Übertragung von Fehlercodes und die Datenlänge entspricht der maximalen Datenlänge eines CAN-Telegramms.

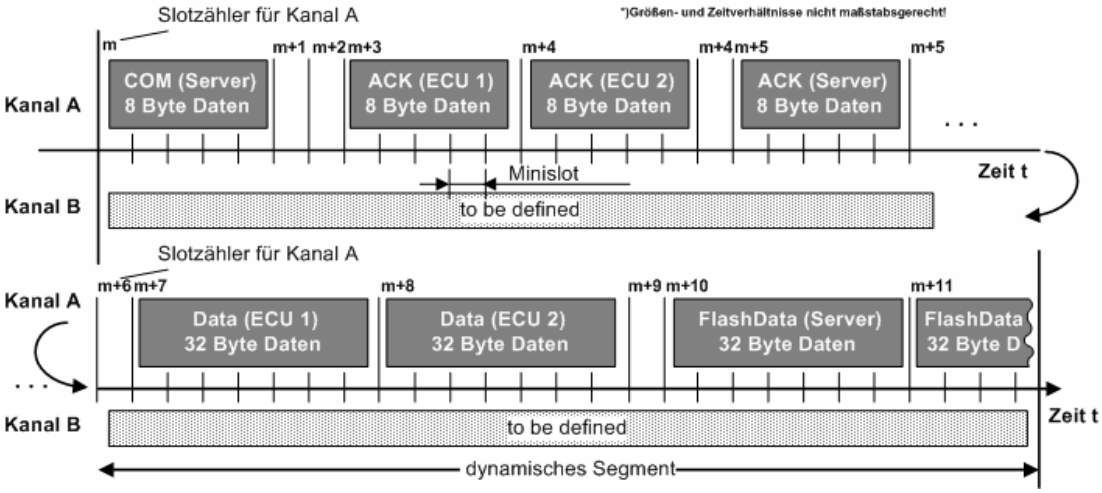

Abbildung 51: Realisierung der Diagnosekommunikation im dynamischen Segment

Die Umsetzung der Diagnosekommunikation innerhalb des Labornetzwerks ist in Abbildung 51 dargestellt. Zwischen den einzelnen Nachrichten wurden einige Minislots nicht benutzt. Diese stehen für spätere Erweiterungen zur Verfügung, um eventuell Nachrichten höherer Priorität zu versenden. Nach den Quittierungs- und Kommandonachrichten folgen die Datennachrichten, die sowohl vom Diagnose-Server als auch vom Steuergerät gesendet werden können. Ein Steuergerät würde beispielsweise den Inhalt seines Fehlerspeichers in solch einem Datenpaket zum Diagnose-Server verschicken. Für die Framelänge der Datennachrichten wurde die maximal mögliche Datenlänge des Prototypen-Clusters von 32 Byte gewählt. Für die Datenpakete der Steuergeräte ist eine Nachricht mit dieser Länge sicherlich ausreichend. Bei den Datenpaketen, die für die Übertragung von Flashdaten genutzt werden, würde eine Erhöhung der Framelänge die Übertragungseffizienz verbessern und die Bandbreite besser ausnutzen. Auf Grund der Limitierung des eingesetzten

FlexRay-Clusters konnten hierzu keine weiteren Untersuchungen durchgeführt werden.

Nach den möglichen Datenpaketen der einzelnen Steuergeräte folgen die Nachrichten mit den Flashdaten vom Diagnose-Server. Diese Übertragung wurde mit drei Nachrichten pro Kommunikationszyklus innerhalb des Labornetzwerks durchgeführt. Der Grund lag in der limitierten Anzahl verfügbarer Puffer der Prototypen. Für spätere Realisierungen ist eine vollständigere Ausnutzung des dynamischen Teils des Kommunikationszyklus sinnvoller. Es bleibt aber abzuwägen, ob ab einer bestimmten Priorität der dynamische Teil vollständig mit Flashdaten-Nachrichten zu füllen ist oder aber sich auf eine Anzahl möglicher Nachrichten beschränkt wird. Bei der ersten Variante würde die verfügbare Bandbreite optimal genutzt. Allerdings stellt sich die Frage, ob dieser Vorteil nicht durch den erhöhten Verwaltungsaufwand wieder aufgehoben würde. Dieser Aufwand entsteht, wenn bei starkem parallelem Diagnoseaufkommen Nachrichten mit Flashdaten und geringster Priorität nicht mehr in einem Kommunikationszyklus gesendet werden können. Da aber die richtige Reihenfolge der Flashdaten beim Steuergerät essentiell ist, müsste diese aufwendig auf dem Diagnose-Server sichergestellt werden.

Ein weiterer Freiheitsgrad bei der Realisierung der Diagnosekommunikation auf einem FlexRay-Backbone ist der zweite Kanal. Dieser könnte neben der Redundanz und Bandbreitenerhöhung auch zur Umsetzung paralleler Diagnosesitzungen genutzt werden. Im Beispiel der Abbildung 51 würde dann die Kommunikation vom Steuergerät 2 (ECU 2) über den Kanal B abgewickelt. Diese Möglichkeit wird am Labornetzwerk in Ermangelung verfügbarer Datenpuffer nicht weiter untersucht. Die gesamte Diagnosekommunikation wird, wie in Abbildung 51 skizziert, über den Kanal A realisiert.

5.3 CAN-Applikation

Zur Repräsentation abgeschlossener Domänen wurden zwei CAN-Netzwerke aufgebaut, wie in Abschnitt 5.1.2 dargestellt [Detering04]. Sie bilden virtuell die Fensterheberfunktionalität von Türsteuergeräten nachen. Zur Steuerung der Fenster verfügen die drei Steuergeräte, die den „Beifahrer“ (BF), „Hinten links“ (HL) und „Hinten rechts“ (HR) symbolisieren, über einen Schalter, mit dem sie ihr virtuelles Fenster zwischen 0 % und 100 % verfahren können. Das vierte Steuergerät stellt ein Fahrer-Türsteuergerät dar. Es verfügt als einziges Steuergerät über vier Schalter, mit denen das eigene virtuelle Fenster direkt und die der anderen Steuergeräte über den CAN-Datenbus gesteuert werden können. In der Verarbeitungslogik sind dabei die Anforderungen für reale Steuergeräte berücksichtigt. So wird bei der Bedienung von zwei Schaltern zum Senken des Fensters das Fenster nach Loslassen des zuerst betätigten Schalters gestoppt. Ein Gegentasten des zweiten Schalters in Richtung Heben wird ignoriert. Außerdem übersteuert der Schalter zum Senken des Fensters in jedem Fall den Befehl zum Schließen des Fensters. Weitere Visualisierungsmöglichkeiten ergeben sich über die angeschlossene LED. Diese kann dazu genutzt werden, das Betätigen des Schalters oder aber einen bestimmten Fensterstatus (offen oder geschlossen) zu visualisieren.

	Softwareversion 1	**Softwareversion 2**
LED	AN – Schalter betätigt AUS – Schalter „in Ruhe“	AN – Schalter „in Ruhe“ AUS – Schalter betätigt
Schalterstellung	LINKS – Fenster senken RECHTS – Fenster heben MITTE – Ruhestellung	LINKS – Fenster heben RECHTS – Fenster senken MITTE – Ruhestellung
LCD – normale Anzeige	Fensterstellung in Prozent	Fensterzustand offen - geschlossen
LCD – Versionsanzeige	„Version 1“	„Version 2“

Tabelle 8: Steuergeräte-Verhalten der unterschiedlichen Softwareversionen

Bei Empfang einer bestimmten CAN-Diagnosenachricht zeigt das Steuergerät für drei Sekunden auf dem LCD seine aktuelle Softwareversion an. Es existieren zwei Versionen, deren Unterschiede in Tabelle 8 dargestellt sind. Mit Hilfe des CAN-Netzwerks soll die Möglichkeit der Update-Programmierung von Steuergeräten über eine Backbone-Architektur gezeigt werden. Die eigentliche Funktionalität der Türsteuergeräte-Applikation ist dabei von untergeordnetem Interesse. Die beiden Softwareversionen können wechselseitig geflasht werden und über das unterschiedliche Verhalten lässt sich die erfolgreiche Programmierung überprüfen. Zusätzlich besteht durch CAN-Diagnosenachrichten die Möglichkeit, Informationen der aktuellen Programm- und Bootloaderversionen über den CAN-Datenbus zu kommunizieren und diese auf dem Diagnosetester zu visualisieren.

5.3.1 Bootloader-Flash (BLF)

Eine Besonderheit der CAN-Steuergeräte stellt der notwendige Bootloader dar, der parallel und getrennt von der eigentlichen CAN-Applikation als eigenständige Software gespeichert ist. Der BLF ermöglicht die Update-Programmierung über den CAN-Bus und stellt eine vollständige Eigenentwicklung dar, deren Einzelheiten im Abschnitt 6.1 vorgestellt werden. Wird ein CAN-Steuergerät angeschaltet, so ist zuerst der BLF aktiv, der das Steuergerät auf Vorhandensein einer gültigen Applikationsversion überprüft. Ist dies der Fall, wird die Applikation gestartet. Befindet sich keine gültige Applikation auf dem Steuergerät, wartet der BLF auf eine Flash-Anforderung. Der BLF ist stark Mikrocontroller-spezifisch und direkt für den C167 entwickelt. Da in dem eingesetzten heterogenen Kommunikationsnetzwerk ausschließlich C167-Mikrocontroller verwendet werden, verfügen alle Steuergeräte mit dem C167 über diesen BLF.

5.4 LIN-Applikation

Zur Komplettierung des heterogenen Kommunikationsnetzwerks wurden vier weitere Steuergeräte entwickelt, die mit einem LIN-Datenbus vernetzt sind (Abschnitt 5.1.3). Die Applikation, die innerhalb des LIN-Netzwerks abläuft, soll ähnlich der CAN-Applikation einen möglichst realitätsnahen Bezug zu einem Kraftfahrzeug herstellen. Darüber hinaus stand eine gute Visualisierbarkeit im Vordergrund, so dass sich für eine optische Funktionalität entschieden wurde [SchadeF04]. Einer der LIN-Slaves (Slave 1) übernimmt die Aufgabe einer

automatischen Innenraumbeleuchtung und steuert zwei LEDs an. Die Ansteuerung kann sowohl lokal am Steuergerät erfolgen als auch über den LIN-Datenbus kommuniziert werden. Ein anderer LIN-Slave (Slave 2) überwacht diese LEDs mittels zweier Fotodioden. Dadurch ist dieser LIN-Slave in der Lage, zu entscheiden, welche der beiden LEDs zurzeit aktiv sind. Der C167 des ersten LIN-Slave wäre natürlich in der Lage, die Überwachungsaufgaben des zweiten LIN-Slave mit zu übernehmen. In diesem Fall ist dies allerdings nicht gewünscht, sondern die Überwachung soll über eine LIN-Kommunikation realisiert werden. In der Realität bestehen intelligente Sensoren und Aktoren zumeist nicht aus solch leistungsfähigen Mikrocontrollern wie dem C167, sondern verfügen eher über eine sehr einfache und kostengünstige Logik.

Der dritte LIN-Slave steuert eine 7-Segment-Anzeige an, mit deren Hilfe die Leuchtdauer in Sekunden von jeweils einer der beiden LEDs visualisiert werden kann. Der vierte LIN-Slave ist gleichzeitig LIN-Master. Er verfügt über Schalter, mit denen alle Aktionen, die an den Slaves lokal vorgenommen werden, vom Master aus kommuniziert werden können. Darüber hinaus kann er ebenfalls über eine eigene 7-Segment-Anzeige den Status des LIN-Netzwerks visualisieren. Zusätzlich wird die Anzeige des Masters genutzt, um die jeweils aktive Softwareversion der LIN-Slaves anzuzeigen.

Ein LIN-Telegramm kann ebenso wie ein CAN-Telegramm maximal acht Nutzdatenbytes enthalten. Für Diagnosezwecke sind spezielle 8 Byte Telegramme im LIN-Protokoll vorgesehen, die auch zum Flashen genutzt werden können. Die Update-Programmierung kann entsprechend analog zum CAN-Datenbus über diese Telegramme erfolgen. Einziger Unterschied ist die im Vergleich zum CAN-Datenbus geringere Datenübertragungsrate, die eine Speicherung der Daten auf dem LIN-Master erforderlich macht. Neue Erkenntnisse ergeben sich aus der Update-Programmierung über den LIN-Datenbus daher nicht, weswegen diese Anwendung hier nicht weiter erörtert werden soll. Wichtiger ist vielmehr die Frage nach dem Routing der Daten zum LIN-Netzwerk und zurück. Dazu wurden Diagnosenachrichten vom Diagnose-Client an das LIN-Netzwerk versandt, die den aktuellen Status des Netzwerks abfragen. Die Antworten der einzelnen LIN-Knoten werden vom Diagnose-Client visualisiert.

5.5 Diagnose-Applikation

Als Diagnose-Applikation wird im Rahmen dieser Arbeit die Off-Board-Kommunikation zwischen Diagnose-Client und Diagnose-Server bezeichnet. Auf die On-Board-Diagnosekommunikation wird in Kapitel 6 weiter eingegangen. Bei dem Diagnose-Server handelt es sich um ein Gateway-Steuergerät, welches im Fahrzeug verbaut ist und die Schnittstelle zur Umwelt zur Verfügung stellt. Ein außerhalb des Fahrzeugs befindlicher Diagnose-Tester repräsentiert einen Diagnose-Client. Wie die Bezeichnungen andeuten, wurde diese Art der Diagnosekommunikation als Client/Server-Architektur realisiert. Dabei stellt ein Server Dienste zur Verfügung, in diesem Fall Diagnosedienste. Ein oder mehrere Clients können sich nach einer möglichen erfolgreichen Authentifizierung mit diesem Server verbinden. Es wird anschließend zwischen Client und Server ein virtueller Kommunikationskanal aufgebaut, über den die Diagnosedienste abgewickelt werden.

Einer der größten Vorteile der Client/Server-Architektur, die im IT-Bereich häufig eingesetzt wird, ist die definierte Schnittstelle zwischen Daten, Ressourcen und Benutzeraktionen. Sie gewährleistet die Daten- und Zugriffssicherheit und bietet abgegrenzte und eindeutige Dienste an. Solche Dienste können beispielsweise der Auf- und Abbau einer Verbindung zu einem Fahrzeug-Steuergerät sein. Der Server überwacht solche Verbindungen und kann je nach Kontext weitere Dienste zur Verfügung stellen. Zum Beispiel könnten nach dem erfolgreichen Aufbau einer Verbindung der Fehlerspeicher des betreffenden Steuergeräts gelesen und anschließend gelöscht werden. Der Einsatz der Client/Server-Architektur bietet darüber hinaus den Vorteil, dass der Server mehrere Clients gleichzeitig bedienen kann. Damit wären parallele Diagnosesitzungen (Sitzung, engl. Session) möglich, um beispielsweise in der Produktion zur gleichen Zeit den Versionsstand der Steuergeräte innerhalb der Infotainment-Domänes abzufragen, das Motorsteuergerät neu zu programmieren und die Türsteuergeräte zu parametrieren, was Zeit und damit Kosten spart. Da diese parallelen Diagnosesitzungen sich auf unterschiedliche Domänen beziehen, sind auch parallele Programmierungen möglich.

Zukünftige Diagnoseschnittstellen bei Fahrzeugen werden voraussichtlich drahtlos konzipiert, um die dadurch entstehenden Komfort- und Wartungsvorteile nutzen zu können. Aus diesem Grund wurde die Off-Board-Diagnosekommunikation mit dem Labornetzwerk ebenfalls drahtlos realisiert. Als mögliche drahtlose Übertragungsprotokolle stehen W-LAN und Bluetooth zur Verfügung, wobei sich hier für die Realisierung auf Bluetooth festgelegt wurde. Die Gründe, die zur Entscheidung für Bluetooth geführt haben, sind in Kapitel 3.4 dargestellt. An dieser Stelle soll allerdings nochmals darauf hingewiesen werden, dass die hier beschriebene Entscheidung keinerlei Wertung zwischen Bluetooth und W-LAN für die Eignung einer Fahrzeug-Diagnoseschnittstelle sein soll. Beide haben ihre spezifischen Vor- und Nachteile, die jeder Fahrzeughersteller für sich selbst gewichten muss. Die Implementierung trägt diesem Umstand durch ihren modularen Aufbau Rechnung. Dadurch lassen sich die beiden Protokolle sowohl auf dem Diagnose-Client als auch auf dem Diagnose-Server relativ einfach gegeneinander austauschen. Die weitere Beschreibung wird sich allerdings auf Bluetooth beschränken. Dafür wird im nächsten Abschnitt ein kurzer Überblick über den Aufbau dieses Protokolls gegeben. Daran schließen sich die Abschnitte an, die die Umsetzungen des Diagnose-Clients und des Diagnose-Servers beschreiben.

5.5.1 Bluetooth-Protokoll

5.5.1.1 Aufbau des Bluetooth-Protokolls

Das Verständnis für den Aufbau des Bluetooth-Protokolls ist zur Realisierung der Diagnosekommunikation notwendig. Einzelheiten zur Leistungsfähigkeit und allgemeine Informationen über Bluetooth werden dagegen im Kapitel 3.4.5 gegeben. Die Bluetooth-Spezifikation unterteilt sich in vier Funktionsbereiche:

- Der hardwarenahe Teil besteht aus dem Hochfrequenz-Bereich, dem Basisband und dem Linkmanager-Protokoll (LMP). Diese Protokollteile sind Bestandteil des Bluetooth-Controllers und stellen die grundlegende

Bluetooth-Funktionalität her. Sie werden durch den jeweiligen Modulhersteller als Firmware implementiert.

- Der Übergang zwischen Bluetooth-Controller und Hostsystem ist als spezielle Schnittstellenfunktionalität im Host Controller Interface (HCI) umgesetzt.
- Entsprechend zum HCI auf Seite des Bluetooth-Controllers existiert auch auf dem Hostsystem eine Zugriffsebene mit dem Logical Link Control and Adaption Protocol (L2CAP). L2CAP bietet die Kommunikationsgrundlage für die aufgesetzten Hostprotokolle durch die Bereitstellung virtueller Kommunikationskanäle.
- Die Hostprotokolle, die auf dem L2CAP aufsetzen, bieten dem Hostsystem eine definierte Anwendungsschicht. Beispiele für Hostprotokolle sind das Cable Replacement Protocol (RFCOMM), das Telephony Control Protocol (TCS) und das Service Discovery Protocol (SDP).

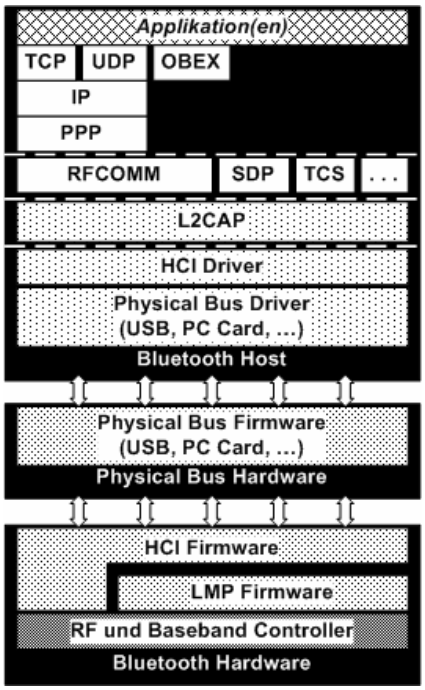

Abbildung 52: Aufbau des Bluetooth-Protokolls (ohne Audio-Layer)

Einen Überblick über diese vier Bereiche gibt die Abbildung 52. Innerhalb des HCI befindet sich meist eine zusätzliche Schnittstelle als externe Hardware, die das Bluetooth-Modul mit dem Hostsystem verbindet. Für den Diagnose-Client und den Diagnose-Server wurde das Bluetooth-Modul über den USB-Bus angebunden.

Oberhalb der Hostprotokolle setzen die adaptierten Protokolle aus anderen Bereichen wie TCP, IP oder PPP auf. Von dieser Möglichkeit wird allerdings bei der Realisierung der Diagnoseapplikation kein Gebrauch gemacht. Diese nutzt vielmehr das RFCOMM-Protokoll, das dem System sowohl synchrone als auch asynchrone virtuelle serielle Schnittstellen zur Verfügung stellt und sich besonders für reine Datenverbindungen eignet.

Je nach Art der Verbindung, symmetrisch oder asymmetrisch, und Fehlerkorrekturmechanismen verwendet Bluetooth bis zu 16 unterschiedliche Pakettypen. Allen gemeinsam ist der grundsätzliche Aufbau, wie er in Abbildung 53 dargestellt ist. Für die Diagnosekommunikation wird eine asymmetrische Verbindung genutzt, die von einem CRC-Code geschützt ist und auf Fehlerkorrekturcodes verzichtet. Damit werden im Hinkanal eine Datenrate von 723,2 kBit/s und im Rückkanal eine Datenrate von 57,6 kBit/s bei optimalen Funkverhältnissen erreicht. Für eine detaillierte Beschreibung der Bluetooth-Datenpakete wird auf [Wollert02] verwiesen.

Abbildung 53: Grundsätzlicher Aufbau des Bluetooth-Datenpaket

Für die Diagnoseanwendung erfolgt die Datenkommunikation transparent durch das Senden und Empfangen von Daten über standardisierte Netzwerk-Schnittstellen. Aus Applikationssicht ist es unerheblich, ob es sich dabei um einen Bluetooth- oder einen W-LAN-Adapter handelt. Für die Absicherung des Diagnose-Servers ist allerdings ein tieferes Verständnis des Bluetooth-Verbindungsaufbaus und der zur Verfügung gestellten Sicherheitsmechanismen notwendig.

5.5.1.2 Verbindungsaufbau und Sicherheit

Für den Aufbau einer Verbindung nutzt Bluetooth die Mechanismen „Inquiry“ und „Paging“. Ersteres kommt zum Einsatz, wenn noch keinerlei Verbindungsinformationen bekannt sind. Durch einen Inquiry-Scan ermittelt ein Bluetooth-Gerät die erreichbaren Geräte innerhalb seiner Reichweite. Stationen, die mit anderen Geräten eine Kommunikation eingehen möchten, hören innerhalb der Dauer eines Inquiry-Scans (1,28 s) in einem Fenster von 11,25 ms, ob ein Inquiry eingeleitet wurde. Nach dem erfolgreichen Empfang antwortet dieses Gerät mit einer Antwort, die alle notwendigen Timing- und Adressinformationen für eine Kommunikation mit ihm enthalten. Der Master wertet diese Antwort aus und registriert das Bluetooth Gerät. Es kann danach durch gezieltes Paging direkt angesprochen werden.

Das Paging bezeichnet den Vorgang, bei dem eine feste Verbindung zwischen Master und Slave aufgebaut wird. Voraussetzung ist, dass der Master die Timing- und Adressinformationen des Slaves besitzt. Analog zum Inquiry-Scan wird beim Paging auf einer Paging-Hopping-Sequenz mit 32 Frequenzen versucht, den Slave zu erreichen. Die Page-Hopping-Sequenz wird aus der 48 Bit langen und weltweit eindeutigen Slave-Adresse abgeleitet und ist damit charakteristisch für einen bestimmten Slave. Dieser antwortet daraufhin auf einer einmalig korrespondierenden Antwortfrequenz mit einem ID-Paket. Daraufhin schickt der Master alle Timing-, Adress- und Hoppinginformationen zum Slave, die dieser benötigt, um sich mit dem Master zu synchronisieren. Der Slave antwortet nochmals und Master und Slave bilden ein Piconet. Die eigentliche Datenübertragung startet anschließend mit einem Poll-Paket, initiiert durch den Master. Danach ist die Verbindung etabliert.

Jede dieser Verbindungen wird durch einen gemeinsamen Verbindungscode zwischen Master und Slave gesichert, der durch eine Zufallszahl, der Bluetooth-

Geräteadresse und einem eindeutigen PIN-Code, der beiden Geräten bekannt sein muss, gebildet wird. Darüber hinaus kann optional der Bluetooth-Kommunikationskanal mit bis zu 128 Bit verschlüsselt werden. Als zusätzliche Sicherheit können die Bluetooth-Geräte dahin gehend konfiguriert werden, dass sie Verbindungen nur mit bestimmten Geräten eingehen. Durch diese in der Protokoll-Spezifikation offen gelegten Sicherheitsmechanismen eignet sich Bluetooth insbesondere bei der Fahrzeugdiagnose für den sicherheitssensitiven Bereich der Programmierung von Fahrzeug-Steuergeräten.

5.5.1.3 Diagnose-Profil

Die Interoperabilität verschiedener Bluetooth-Geräte wird durch Bluetooth-Profile festgelegt, die bestimmten Anwendungen zugeordnet sind. Innerhalb der Profile sind die benötigten Protokollimplementierungen und Ressourcen spezifiziert, die Bluetooth-Geräte besitzen müssen, um bestimmte Dienste anbieten oder nutzen zu können. Damit sich einander unbekannte Bluetooth-Geräte über ihre jeweiligen Fähigkeiten austauschen können, existiert das Service Discovery Protocol (SDP). Dabei muss ein Gerät, das einen bestimmten Dienst zur Verfügung stellt, einen SDP-Server implementieren. Ein Gerät, was einen Dienst nutzen möchte, verfügt über einen SDP-Client.

Für den Diagnosezugang zum Labornetzwerk über Bluetooth wurde aus diesem Grund ebenfalls ein eigenes Diagnose-Profil implementiert. Der Diagnose-Server bietet das „emg-Diagnose“-Profil[30] an. Dabei ist festgeschrieben, dass ein Client über mindestens eine RFCOMM-Implementierung verfügen muss und welche Verbindungsparameter (z.B. Kanalnummer) einzustellen sind. Der Diagnose-Client kann alle entfernten Bluetooth-Geräte nach diesem speziellen Profil abfragen. So wird sichergestellt, dass eine Verbindung zur Diagnosekommunikation nur zwischen Bluetooth-Geräten zustande kommt, die über das Profil „emg-Diagnose“ verfügen. Als weitere Sicherheitsmaßnahmen greifen dann Passwortabfrage und eine eventuelle Verschlüsselung.

5.5.2 Diagnose-Server

Der Diagnose-Server stellt ein Steuergerät dar, das sich innerhalb des Fahrzeugs befindet, eine Bluetooth-Schnittstelle besitzt und am FlexRay-Backbone angeschlossen ist. Im Labornetzwerk wurde dieses Steuergerät durch einen PC repräsentiert, der mit einem echtzeitfähigen Linuxsystem (Debian mit RTAI-Aufsatz) betrieben wird [RTAI] und über einen FlexRay-Anschluss verfügt. Die Bluetooth-Schnittstelle ist per USB-Bus angebunden. Im Folgenden wird die verwendete Echtzeiterweiterung RTAI kurz vorgestellt und danach die realisierte Diagnose-Applikation auf dem Diagnose-Server beschrieben.

[30] „emg-Diagnose“ aus dem Grund, da „emg“ für die Abkürzung des Instituts steht, an dem diese Arbeit entstanden ist.

5.5.2.1 RTAI-Linux

Das Betriebssystem Linux trennt die Laufzeitumgebungen von Prozessen zwischen so genanntem Kernel-Space und User-Space. Innerhalb des User-Space laufen Prozesse geschützt ab und haben keinen direkten Zugriff auf Hardware oder systemkritische Komponenten wie Speicher. Im Kernel-Space dagegen sind direkte Hardware- und Speicherzugriffe erlaubt. Gerätetreiber laufen typischerweise im Kernel-Space, während Nutzer-Applikationen im User-Space ablaufen. Linux ist mit diesem Konzept allerdings nicht echtzeitfähig. Dies wird erst durch den Ansatz erreicht, den RTAI verfolgt und der in Abbildung 54 dargestellt ist. Dabei wird bei RTAI eine zusätzliche schlanke Hardware-Abstraktionsschicht eingeführt, die sich hauptsächlich um externe Ereignisse (Interrupts) und die Ressourcenvergabe unter Beachtung der Echtzeitanforderungen kümmert. Um die Echtzeit garantieren zu können, müssen alle RTAI-Applikationen als Kernel-Module im Kernel-Space ablaufen. Das eigentliche Linux-System läuft ebenfalls als RTAI-Task mit geringster Priorität parallel zu den anderen RTAI-Tasks im Kernel-Space.

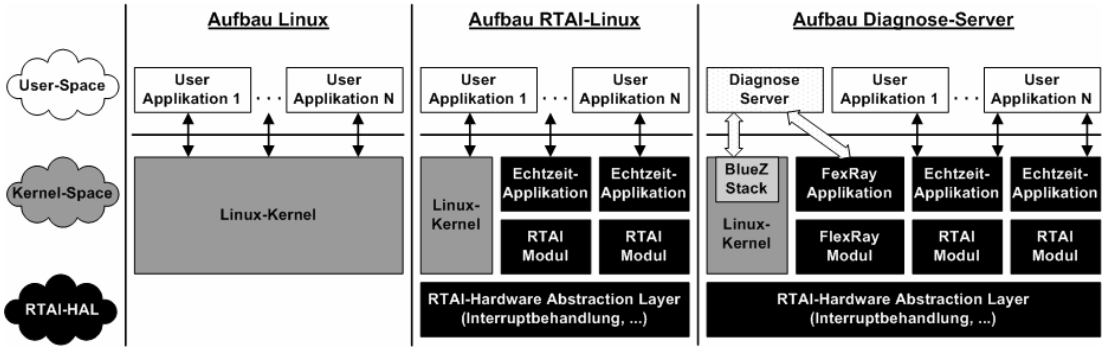

Abbildung 54: Aufbau der zum Einsatz kommenden Betriebssystem-Kernel für Linux, RTAI-Linux und im Diagnose-Server

Da unter Linux bzw. RTAI jeder Prozess seinen eigenen geschützten Speicherbereich besitzt, muss für den Datenaustausch der Prozesse untereinander eine Interprozess-Kommunikation (IPC) zum Einsatz kommen. RTAI unterstützt als IPC-Mechanismen FIFO-Puffer, Pipes und Shared Memory mit den jeweils dafür benötigten Signalisierungsmechanismen wie Mutexe und Semaphoren. Zur Erklärung dieser Begriffe wird auf die Literatur verwiesen. Für den Diagnose-Server wurden für die Interprozess-Kommunikation Semaphoren-geschützte FIFO-Puffer und Shared Memory eingesetzt.

5.5.2.2 Diagnose-Applikation

Wie die Abbildung 54 zeigt, läuft die vollständige Diagnose-Applikation im User-Space als eigenständiges Programm ab. Die Hauptaufgabe der Diagnose-Server-Anwendung ist dabei die Abwicklung der vollständigen Bluetooth-Kommunikation. Dazu installiert der Server bei der Initialisierung das „emg-Diagnose“-Profil und überwacht die Bluetooth-Schnittstelle auf Anfragen von erlaubten Bluetooth-Geräten.

Verbindet sich ein Diagnose-Client mit dem Diagnose-Server, so muss der Nutzer sich über ein Passwort beim Diagnose-Server authentifizieren. Danach ist die Diagnosesitzung etabliert. Durch die Client/Server-Architektur und die Möglichkeit von Bluetooth mit mehreren Geräten parallel kommunizieren zu können, ist der Diagnose-Server in der Lage, auch gleichzeitig verschiedene Diagnosesitzungen zu behandeln. Um die Anforderungen an den Diagnose-Server zu begrenzen, werden die parallelen Diagnosesitzungen per Software auf maximal drei beschränkt. Eine Einschränkung bei parallelen Diagnosesitzungen besteht darin, dass ein Steuergerät nur von einem Client adressiert werden darf. Ansonsten kann der Server eventuelle Antworten des Steuergeräts nicht zuordnen, da dieses in dem hier implementieren Diagnose-Protokoll keinen bestimmten Diagnose-Client adressieren kann.

Die Daten, die der Diagnose-Server über Bluetooth vom Client erhält, werden auf Vollständigkeit und Korrektheit überprüft. Der Telegrammaufbau ist mit einer Länge von acht Bytes an den Aufbau der CAN-Telegramme angelehnt. Sie bestehen aus einem Header, der die Adresse des zu adressierenden Steuergeräts und den Code des Diagnose-Service enthält, sowie den eigentlichen Daten. Diese können Parameter, Fehlercodes oder Werte enthalten. Den gleichen Aufbau besitzen die Telegramme, die für die Update-Programmierung genutzt werden. Einzig die Länge nutzt die maximale Datenlänge der Bluetooth-Datenpakete aus, um den Datendurchsatz möglichst hoch werden zu lassen und die Daten schnell „ins Fahrzeug“ zu bekommen.

Die Daten, die direkt für den Server bestimmt sind, werden in einem gesonderten Task von diesem verarbeitet. Alle anderen Daten, die Steuergeräte im Fahrzeug adressieren, werden über den FlexRay-Backbone weiter geleitet. Dazu bedient sich der Server der IPC und tunnelt die Daten in den Kernel-Space. Die dort aktive FlexRay-Applikation empfängt die Daten und sendet sie über die dafür vorgesehenen FlexRay-Slots in das Kommunikationsnetzwerk. Daten, die als Antwort von Steuergeräten zum Diagnose-Client gesendet werden, werden ebenfalls über IPC-Mechanismen zurück in den User-Space getunnelt und von der Diagnose-Applikation empfangen. Diese sendet die Antwort per Bluetooth an den Diagnose-Client.

5.5.3 Diagnose-Client(s)

Bei der Applikation, die auf den Diagnose-Clients abläuft, handelt es sich im Wesentlichen um ein Standardprogramm, das einen Bluetooth-Stack nutzt und repräsentiert innerhalb des Labornetzwerks einen möglichen Diagnose-Tester. Realisiert wurde diese Applikation auf einem Notebook unter dem Betriebssystem Linux, das als Debian-Distribution in der Version „Sarge“ und der Kernelversion 2.4.27 zum Einsatz kam [Debian]. Das kostenlose Betriebssystem Linux bot sich für diese Entwicklungsaufgabe an, da neben einer großen Anzahl nutzbarer Werkzeuge der ebenfalls frei verfügbare leistungsfähige Bluetooth-Stack „BlueZ“ unterstützt wird [BlueZ]. Zudem kommt Linux auch auf dem Diagnose-Server und als Embedded Version auf den FlexRay-Knoten zum Einsatz.

Die Diagnose-Applikation des Clients ist modular aufgebaut und die eigentliche Diagnose-Funktionalität von der Benutzerinteraktion getrennt. Dies ermöglicht die

einfache Portierbarkeit auf andere Systeme. So wurden für das Labornetzwerk eine Diagnose-Applikation als reine Konsolenanwendung und als eine grafische Benutzeroberfläche unter Qt realisiert. Exemplarisch wurden dabei einige Funktionen eines Diagnose-Testers implementiert, die in der Tabelle 9 aufgelistet sind.

Um diese Funktionen nutzen zu können, muss zuerst nach einem erreichbaren Diagnose-Server gesucht werden, der über das „emg-Diagnose“-Profil verfügt. Ist dieser gefunden und hat der Benutzer sich erfolgreich authentifiziert, kann ein Steuergerät gewählt werden, mit dem die weitere Diagnose-Kommunikation stattfinden soll. Die Realisierung mit dem Labornetzwerk geht dabei von bekannten und vorhandenen Steuergeräten aus. Als Verbesserung könnte für diesen Zweck ein Mechanismus entwickelt werden, der innerhalb des Kommunikationsnetzwerks nach vorhandenen Steuergeräten sucht und diese Informationen dem Diagnose-Client zur Verfügung stellt.

Aktion	Beschreibung
„Diagnose-Server suchen“	Sucht nach einem Diagnose-Server mit dem „emg-Diagnose“-Profil
„Diagnosesitzung öffnen“	Baut eine Verbindung auf und benötigt eine Authentifizierung
„Steuergerät wählen“	Zeigt eine Liste vorhandener Steuergeräte
„Fehlerspeicher lesen“	Liest den Fehlerspeicher des ausgewählten Steuergeräts
„Fehlerspeicher löschen“	Löscht den Fehlerspeicher des ausgewählten Steuergeräts
„Status abfragen“	Fragt nach Softwareversionen auf dem ausgewählten Steuergeräts
„Update-Programmierung durchführen“	Fragt nach der zu programmierenden Software und programmiert damit das ausgewählte Steuergerät
„Diagnosesitzung schließen“	Beendet die Verbindung mit dem Diagnoseserver

Tabelle 9: Implementierte Diagnose-Tester Funktionalitäten

6 Update-Programmierung im heterogenen Kommunikationsnetzwerk

Mit dem im vorherigen Kapitel beschriebenen heterogenen Backbone-basierten Kommunikationsnetzwerk wurde eine Domänen-interne und Domänen-übergreifende On-Board-Kommunikation realisiert und die Funktionsfähigkeit nachgewiesen. Dabei ist bislang die Einschränkung der Backbone-Architektur einer notwendigen Vermittlungsschicht zur Steuergeräte-Adressierung nicht weiter berücksichtigt. Ließe sich die Domänen-übergreifende Standard-On-Board-Kommunikation zwischen einzelnen Steuergeräten noch relativ einfach über statische Routingtabellen bewerkstelligen, so benötigt die sporadische und temporäre Off-Board-Diagnosekommunikation einen flexibleren Ansatz.

Als Funktionsnachweis für das in diesem Kapitel vorgestellte Routingprotokoll werden über die Diagnosekommunikation Steuergeräte des Netzwerkverbundes von „außen“ geflasht. Eine der wichtigen Voraussetzungen für die Umprogrammierung eines Steuergerätes über den angeschlossenen Datenbus stellt der Bootlader-Flash (BLF) dar. Im folgenden Abschnitt werden der Aufbau und die Funktionsweise eines BLF beschrieben und Erkenntnisse für den Einsatz im Fahrzeug abgeleitet. Anschließend werden das entwickelte Routingprotokoll und die dahinter stehenden Überlegungen vorgestellt.

6.1 Bootloader-Flash (BLF) für Steuergeräte

6.1.1 Funktionsweise eines Bootloader-Flash (BLF)

Der Bootloader-Flash befindet sich parallel zur Applikation auf jedem flashbaren Steuergerät und ermöglicht als eigenständige Anwendung die (Re-)Programmierung des Steuergeräts. Das Programmieren, was den teilweisen oder gesamten Austausch der Steuergeräte-Applikation umfassen kann, erfolgt im eingebauten Zustand über einen angeschlossenen Datenbus wie beispielsweise dem CAN-Bus. Dadurch ist es möglich, den Programmierablauf über den gesamten Lebenszyklus des Steuergeräts (Entwicklung, Produktion, Kundendienst) beizubehalten.

Die Übertragung der Daten zum Steuergerät erfolgt mittels standardisierter Transport- und Diagnoseprotokolle, für die der Bootloader entsprechende Treiber benötigt. Die herstellerabhängigen Flash-Routinen, die zur Programmierung erforderlich sind, können, wenn sie nicht direkt im BLF enthalten sind, über den Datenbus transportiert werden. Bevor die Programmierung beginnen kann, muss der BLF die Authentizität der Software überprüfen. Dies beinhaltet den Schutz vor Manipulation, den Schutz vor Programmierung unrechtmäßig angefertigter Kopien und eine Versionskontrolle (Austausch der Software nur gegen aktuelle Versionen). Dazu wird die Software durch den Automobilhersteller mit einer kryptographischen Signatur versehen, mit der die Integrität der Software gewährleistet werden kann. Des Weiteren stellt der BLF über ein Challenge-Response- oder ein Public-Key-

Verfahren sicher, dass die Programmierung ausschließlich über ein autorisiertes Programmiergerät erfolgt.

Sind die Sicherheitsüberprüfungen erfolgreich verlaufen, kann mit dem Download der Daten über den Datenbus begonnen werden. Dazu erhält der BLF Informationen über den zu beschreibenden Speicherbereich, der daraufhin gelöscht wird. Hier muss sichergestellt sein, dass kein Speicherbereich überschrieben wird, der vom BLF genutzt wird.

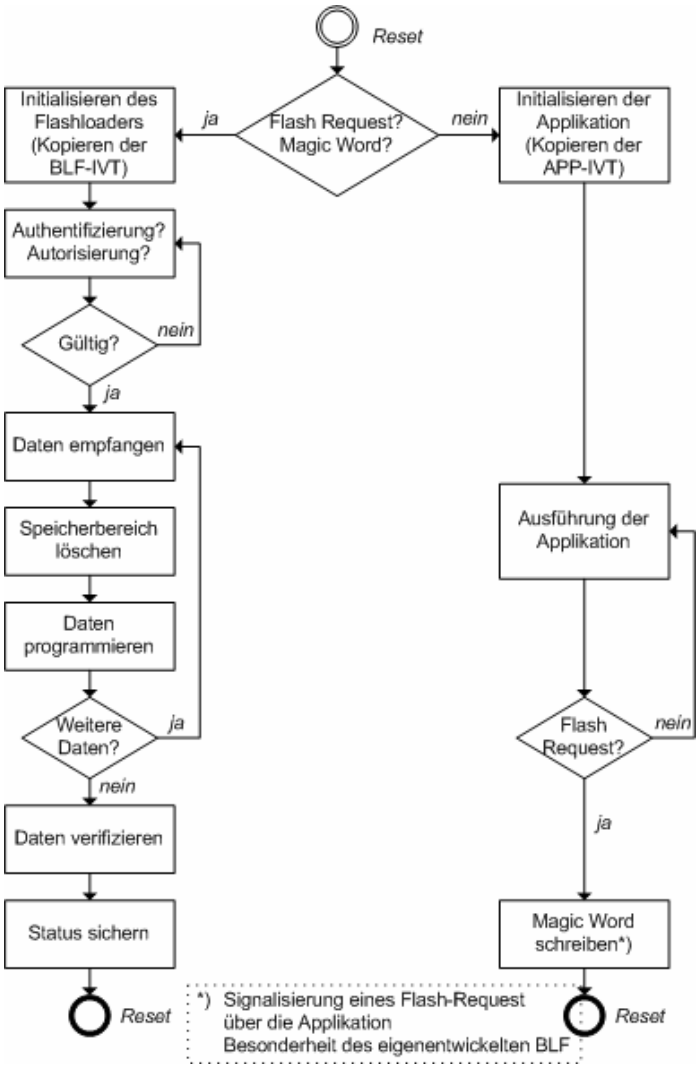

Abbildung 55: Flussdiagramm für das Ablaufprinzips eines Flashloaders

Sind alle Daten geschrieben, überprüft der BLF mittels Signatur- und Checksummenverfahren die Integrität der Software. Ist diese in Ordnung, wird der aktuelle Flash-Status geschrieben und die Applikation gestartet. Anderenfalls verbleibt der BLF im Programmiermodus. In Abbildung 55 ist dieser prinzipielle Ablauf eines Bootloader-Flashs in einem Flussdiagramm dargestellt.

6.1.2 Anforderungen an einen Bootloader-Flash (BLF)

Der Bootloader-Flash muss gewährleisten, dass das Steuergerät bei funktionsfähiger Hardware zu jedem Zeitpunkt programmierbar bleibt (Robustheit). Dies gilt insbesondere nach dem planmäßigen oder unplanmäßigen Abbruch eines vorhergegangen Programmiervorgangs oder bei fehlerhafter bzw. fehlender Steuergeräte-Applikation. Durch entsprechende Mechanismen des BLF muss sichergestellt sein, dass Störungen im Programmierablauf nicht zu einem permanenten Ausfall des Steuergeräts führen. Dazu ist es erforderlich, dass die Gültigkeit der Applikation auf dem Steuergerät vom BLF festgestellt und überprüft werden kann. Dies kann mit Hilfe von digitalen Signatur- und Prüfsummenverfahren geschehen. Weitere Anforderungen ergeben sich aus dem eigentlichen Programmiervorgang, der stark systemspezifisch ist. Allerdings lassen sich prinzipielle Aussagen treffen. Dafür sind ein tieferes Verständnis zur Funktion des Flash-Bootloaders (BLF) und einige Details zur Arbeitsweise bei Mikrocontrollern notwendig.

Eine der wesentlichen Eigenschaften von Prozessoren ist das Reagieren auf externe Ereignisse mittels Interrupts. Dazu ist jede mögliche Quelle, die einen Interrupt auslösen kann, wie z.B. Timer oder die serielle Schnittstelle, fest einem bestimmten Speicherbereich zugeordnet. Der Speicherbereich umfasst eine Speicheradresse mit einer festen zugehörigen Länge und enthält den der Quelle zugeordneten Interruptvektor. Bei einem Interruptvektor handelt es sich um einen Sprungbefehl an die Speicheradresse der Interruptserviceroutine (engl. Interrupt Service Routine, ISR). Der Speicherbereich mit den Interruptvektoren wird Interruptvektor-Tabelle (engl. Interupt Vector Table, IVT) genannt und beginnt meist bei der Speicheradresse Null (0000h). Im Fall des eingesetzten C167 besitzt die IVT eine Größe von 512 Bytes (hexadezimal 200h). Nach dem Einschalten oder nach einem Reset befindet sich der Mikrocontroller in seinem Grundzustand, d.h. sein Progammzähler zeigt auf die Speicheradresse 0000h. Dort befindet sich ein Interruptvektor, der auf die Einsprungadresse des sich im Flashspeicher befindlichen Programmcode zeigt und auch als Reset-Vektor bezeichnet werden kann. Durch den Sprung an diese Adresse, die sich im Reset-Vektor befindet, wird das Programm gestartet.

Wird ein BLF neben der eigentlichen Applikation, die auch als Fahrprogramm (FP) bezeichnet wird, auf dem Mikrocontroller eingesetzt, so besteht das Problem darin, dass sich zwei voneinander unabhängige Anwendungen die Ressourcen des Mikrocontrollers teilen müssen. Jede dieser beiden Applikationen verfügt beispielsweise über ihre eigene IVT, die mit der IVT des Mikrocontrollers verknüpft werden muss. Darüber hinaus befinden sich beide Applikationen im gleichen Flash-Speicher des Mikrocontrollers. Hier muss sichergestellt sein, dass ein notwendiges Löschen des Speichers für eine Update-Programmierung den BLF-Code unangetastet lässt. Durch den Aufbau und die Funktionsweise heutiger Mikrocontroller stellt gerade diese Forderung eine große Herausforderung für die Entwicklung eines BLF dar.

Bei jeder Neuübersetzung der Software (Kompilierung) besteht die Möglichkeit, dass der Programmcode und seine ISR verschiedenen Speicherbereichen zugeteilt werden (Mapping). Damit ändert sich gleichzeitig die IVT des Programms und muss danach ebenfalls neu geflasht werden, d.h. der Speicherbereich ab Adresse 0000h wird neu geschrieben. Das Problem dabei ist die derzeit eingesetzte Flashspeichertechnologie, die nicht adress- sondern nur segmentweise gelöscht werden kann. Die Größe eines

solchen Segments hängt von der Organisation der Speicherbausteine und der Anbindung mit Adressleitungen ab und liegt im Kilobyte-Bereich. Dies bedeutet, dass damit zwangsläufig der IVT des Controllers gelöscht wird, der nur einige hundert Bytes umfasst. Gleichzeitig würde dies die IVT des Bootloaders unbrauchbar machen. Sollte während des Löschvorgangs oder vor dem Beschreiben einer neuen IVT ein Fehler auftreten, kann der Controller nur noch über seine spezielle Programmierschnittstelle programmiert werden. Aus diesem Grund muss ein BLF zu jedem Zeitpunkt funktionsfähig bleiben. Daraus ergeben sich zwei Forderungen:

- Getrennte Flash-Speicherbereiche für Bootloader und Fahrprogramm
- One-Time-Programming für den BLF-Flash-Speicherbereich

Der Einsatz von getrennten Speicherbereichen würde das versehentliche Beschädigen des BLF durch fehlerhafte Flash-Routinen oder menschliche Fehlbedienung sicher verhindern. Solch ein Flash-Speicher benötigt nur wenige Kilobytes und würde sehr elegant das Problem der verschiedenen Interrupt-Vektortabellen lösen. Der Mikrocontroller könnte vom Design so ausgelegt sein, erst den BLF mit der dazugehörigen IVT zu starten und danach, bei einem Wechsel zum Fahrprogramm, dessen IVT zu aktivieren. Darüber hinaus umgeht diese Vorgehensweise den Nachteil einer großen Anzahl von Mikrocontrollern, die nicht gleichzeitig den Flash-Speicher löschen können, aus dem sie gerade ein Programm ausführen. Daher muss bislang zum Löschen des Flash-Speichers der Code des BLF erst in den RAM kopiert und von dort ausgeführt werden, auch wenn der Speicherbereich des BLF direkt nicht gelöscht würde. Der RAM-Bereich muss somit die Codegröße des BLF, zumindest der Routinen für „Flash-Erase“ und „Flash-Program“, aufnehmen können und zusätzlich noch Platz für die Zwischenspeicherung der zu programmierenden Daten bereitstellen. Die Nutzung getrennter Flash-Speicher würde den Bedarf an RAM verringern bzw. den verfügbaren RAM besser ausnutzen.

Die Möglichkeit, den Flash-Speicher des BLF nur einmalig programmieren zu können, verhindert ebenfalls sicher unbeabsichtigtes und unrechtmäßiges Löschen des Bootloaders. Darüber hinaus bietet sich dieser Speicher zusätzlich zur Speicherung von digitalen Signaturen an, die der BLF zur Integritätsprüfung und zur Authentifzierung nutzen kann.

6.2 Entwicklung eines BLF für die Laborsteuergeräte

Die in vorangegangenen Abschnitten beschriebenen Anforderungen für einen Bootloader-Flash galt es für die eingesetzten C167-Mikrocontroller umzusetzen. Diese verfügen jeweils über Flash-Speicher mit einer Größe von 256 kByte und RAM-Speicher, ebenfalls mit einer Größe von 256 kByte. Durch die Architektur des Mikrocontrollers werden diese beiden Bereiche linear hintereinander adressiert. Die verwendeten Flash-Speicherbausteine sind in 16 kByte großen Blöcken organisiert, die nur komplett gelöscht und anschließend beschrieben werden können. Für den Einsatz des Bootloaders wurden die ersten beiden 16 kByte Blöcke reserviert, die gleichzeitig die IVT des Mikrocontrollers enthalten. Im ersten der beiden Blöcke ist der BLF-Programmcode abgelegt und im anderen wird der aktuelle Flash-Status gespeichert. Die Applikation kann somit die restlichen 14 Blöcke mit einer Gesamtgröße von 224 kByte nutzen (Adressbereich ab 8000h). Durch die

eingesetzten Flash-Routinen wird programmtechnisch sichergestellt, dass keine Adressbereiche unterhalb von 8000h gelöscht oder beschrieben werden. Gegen Fehler in der Implementierung oder Ausnutzen von Softwarefehlern (Hacking) hilft dieser Schutz allerdings nur bedingt.

6.2.1 Starten des Flash-Prozesses

Die wichtigste Anforderung, die es zu erfüllen gilt, ist die Sicherstellung der Bootloaderfunktion zu jedem Zeitpunkt. Dazu musste die im vorherigen Abschnitt beschriebene Problematik der Interrupt-Vektortabellen gelöst werden. Einen Ansatz, um dies bei gebräuchlichen Mikrocontrollern zu realisieren, soll anhand der Abbildung 56 näher erläutert werden.

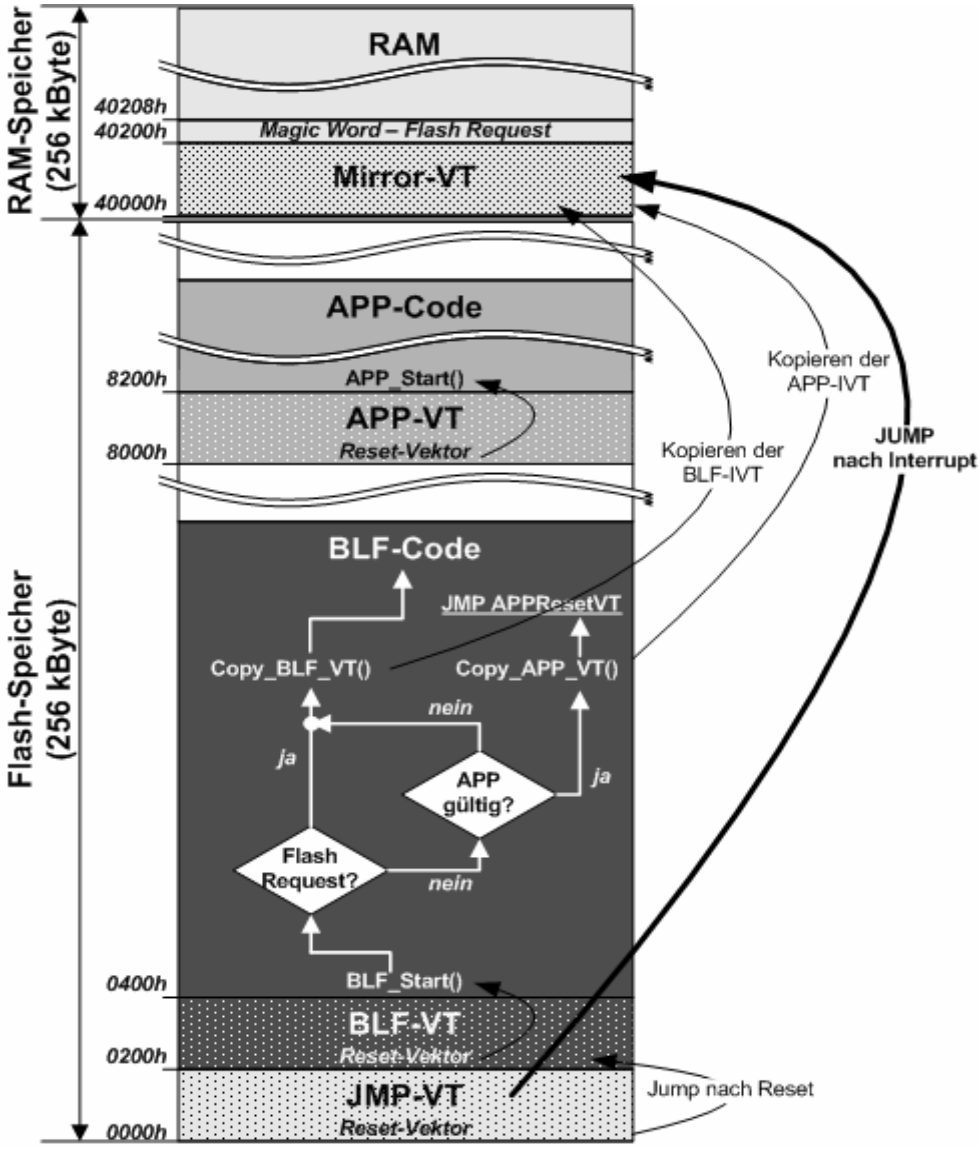

Abbildung 56: Prinzip der IVT-Umschaltung des eigenentwickelten Flashloaders

Die Standard-IVT des Mikrocontrollers befindet sich an Speicheradresse 0000h, bei der der Reset-Vektor auf die Einsprungadresse des Programms zeigt, und liegt im

gleichen Speicherblock wie der Programmcode des BLF. Damit lassen sich die Daten in diesem Bereich entsprechend den Anforderungen nachträglich nicht mehr verändern. Um dennoch die Interruptvektoren an aktuelle Applikationsversionen anpassen zu können, bedarf es eines „Kunstgriffs“. Dazu wird im RAM-Speicher ein Bereich für einen Mirror-IVT (dt. Spiegel) reserviert (siehe Abbildung 56). Der Standard-IVT enthält statische Sprungadressen zu den entsprechenden Adressen des Mirror-IVT mit einer Ausnahme. Der Reset-Vektor verweist auf den Reset-Vektor des BLF-IVT. Damit wird nach jedem Reset zum Reset-Vektor des BLF gesprungen, der wiederum den BLF startet. Alle anderen Interrupts werden zur Mirror-IVT in den RAM geleitet.

Der BLF prüft beim Starten ein „Magic Word“ im RAM, ob dieses einen speziell vereinbarten Wert besitzt. Ist dies der Fall, so bedeutet dies, dass die Steuergeräte-Applikation eine Flash-Anforderung (engl. Request) signalisiert. Der BLF löscht diese Signalisierung und bearbeitet die Flash-Anforderung. Liegt keine Anforderung vor, so entscheidet der BLF anhand der Statuseinträge des zweiten Flash-Speicherblocks, ob eine gültige Applikation vorhanden ist. Wenn nicht, wartet der BLF auf eine Flash-Anforderung zur Programmierung einer Applikation. Befindet sich dagegen eine gültige Applikation im Flash-Speicher, kopiert der BLF die IVT der Applikation, die an einer bekannten Stelle im Speicher abgelegt ist, in den RAM-Bereich des Mirror-IVTs. Anschließend wird durch einen Sprung zum Reset-Vektor der Applikation der BLF deaktiviert und die Applikation gestartet. Diese ist nun vollständig aktiv und kann alle Ressourcen des Mikrocontrollers nutzen. Die einzige Einschränkung durch das Vorgehen mittels Mirror-IVTs liegt in der verlängerten Latenzzeit durch den zusätzlichen Sprungbefehl bei der Bearbeitung von Interrupts. Beim verwendeten C167 mit 20 MHz Taktfrequenz beträgt die Latenzzeit 100 µs (2 Takte). Für noch höher getaktete Mikrocontroller sinkt die Latenzzeit entsprechend und kann nahezu vernachlässigt werden.

Empfängt die Steuergeräte-Applikation über eine Diagnosenachricht eine Flash-Anforderung, so wird die Applikation in einen sicheren Zustand versetzt und das „Magic Word“ in den RAM-Speicher geschrieben. Anschließend führt die Applikation einen Reset aus, der den BLF aktiviert und den Programmierprozess in Gang setzt.

6.2.2 Flash-Programmierung durch den BLF

Wird anstelle der Applikation der BLF aktiviert, sei es auf Grund einer fehlenden Steuergeräte-Applikation oder einer Flash-Anforderung, so wird der IVT des BLF in den Mirror-IVT im RAM-Bereich kopiert. Danach ist der BLF vollständig als eigenständige Applikation aktiv und wartet auf eine erneute[31] Flash-Anforderung. Anschließend müssen die herstellerspezifischen Sicherheitsprüfungen wie Authentifizierung und Autorisation durchgeführt werden, die im Rahmen des Labornetzwerks nicht weiter implementiert wurden. Des Weiteren sind Mikrocontroller-abhängige Besonderheiten zu berücksichtigen. So verhindert eine Vielzahl von Mikrocontrollern das Löschen von Speichern, aus denen das Programm ausgeführt wird.

[31] Erneute Flash-Anforderung deshalb, da vom Programmiergerät immer zwei Flash-Anforderungen geschickt werden müssen. Dies liegt an dem Konzept des BLF, der nicht direkt aus der Steuergeräte-Applikation gestartet werden kann.

Aus diesem Grund kopiert der BLF seinen eigenen Programmcode in das RAM und spiegelt sich dort selbst. Der BLF kann danach vollständig aus dem RAM ablaufen und benötigt keine Zugriffe mehr auf seinen Programmcode im Flash-Speicher. Anschließend kann mit der Programmierung begonnen werden.

Der BLF richtet einen Ringpuffer für die zu programmierenden Daten ein, die fortwährend empfangen werden können. Sollte der Datenpuffer einen kritischen Füllgrad erreichen, so ist der BLF in der Lage über das Diagnoseprotokoll den weiteren Datentransfer zu verlangsamen. Bevor jedoch mit der Programmierung der Applikation begonnen werden kann, muss der BLF für einen konsistenten Zustand des Steuergeräts auch in einem eventuell möglichen Fehlerfall sorgen. Dazu löscht der BLF den zweiten Speicherblock des Flash-Speichers, der den aktuellen Flash-Status enthält. Anschließend wird der Status aktualisiert und neu programmiert, d.h. es wird die Anzahl der bisherigen Flash-Versuche um eins erhöht und eine ungültige Steuergeräte-Applikation vermerkt.

Bei den eingesetzten C167-Mikrocontrollern und den dafür entwickelten Steuergeräte-Applikationen im Labornetzwerk war auf Grund der im Vergleich zu den Applikations-Codemengen relativ großen Speicherblock-Organisation nur eine vollständige Programmierung sinnvoll. Daher wird beim ersten Eintreffen von Flash-Daten der vollständige Speicherbereich der Steuergeräte-Applikation gelöscht. Anschließend können die Daten aus dem Datenpuffer in den Flash-Speicher programmiert werden. Sind keine Daten mehr vorhanden und signalisiert das Programmiergerät das vollständige Ende des Downloads, bildet der BLF über den Programmcode eine Prüfsumme. Diese Prüfsumme wird mit der vom Programmiergerät gelieferten Prüfsumme verglichen. Sind die beiden Prüfsummen unterschiedlich, so ist von korrumpierten Daten auszugehen und die Applikation muss erneut geflasht werden.

Wird dagegen über die Prüfsummen die korrekte Integrität des Applikationscodes festgestellt, aktualisiert der BLF anschließend den Status im zweiten Speicherblock. Dazu wird dieser Speicherblock wiederum gelöscht. Die Anzahl der erfolgreichen Flash-Versuche werden um eins erhöht, das aktuelle Datum wird gespeichert und die Applikation als gültig gekennzeichnet. Danach führt der BLF einen Reset aus, worauf nach den in den vorangegangenen Abschnitten beschriebenen Prüfungen die nunmehr aktuelle Steuergeräte-Applikation gestartet wird.

6.3 Kommunikation mit dezentraler Backbone-Architektur

Nachdem das Prinzip einer sicheren und robusten Programmierung der Steuergeräte über einen Bootloader-Flash beschrieben wurde, bleibt die Fragestellung zu klären, wie die Flash-Daten ihren Weg zum jeweiligen Steuergerät finden. Die Kommunikationsnetzwerke heutiger Fahrzeuge verfügen nur über rudimentäre Routingmechanismen. Die logische Adressierung der einzelnen Steuergeräte wird über fahrzeugweit eindeutige ein Byte große Adressen realisiert, die jeder Fahrzeughersteller konzernweit standardisiert hat (z.B. Motorsteuergerät – 01h). Durch die verwendete Gateway-Architektur reicht es bei dieser Vorgehensweise bislang aus,

die für das Routing benötigten Informationen über die Struktur des Kommunikationssystems zentral im Gateway vorzuhalten.

Dazu existieren im zentralen Gateway statische Verbaulisten (welche Steuergeräte sind verbaut) und Verbindungsmatrizen (an welchem Datenbus ist das Steuergerät angeschlossen). Da durch diese Architektur kaskadierte Datenbusse[32] vermieden werden und die damit verwendete Ein-Byte-Adressierung der Steuergeräte eindeutig ist, genügt es, die Daten vom Gateway auf den richtigen Datenbus verteilen zu lassen. Die Kenntnis über die genaue Struktur des Fahrzeug-Kommunikationssystems ist für eine Kommunikation von „außen", beispielsweise für einen Diagnosetester zur Diagnosekommunikation, nicht notwendig.

Durch den in dieser Arbeit vorgeschlagenen Architekturwandel von der zentralen Gateway-Architektur zur dezentralen Backbone-Architektur entfällt mit dem Super-Gateway das Wissen über die Struktur des Kommunikationssystems. Darüber hinaus werden die Daten in der Regel über mindestens zwei Datenbusse, den Backbone sowie den Ziel-Datenbus, transportiert. Für die Realisierung einer solchen bidirektionalen Kommunikation müssen daher die Struktur des Kommunikationssystems im Netzwerk verteilt gespeichert und die Daten mit einer aufwendigeren Adressierung versehen werden.

Als Vorbild zur Kommunikation innerhalb dezentraler Netze kann das Internet als das größte existierende dezentrale Netzwerk dienen. Durch das ISO/OSI-7-Schichtenmodell lässt sich das Internet mit dem automobilen Netzwerk vergleichen. Die im Fahrzeug eingesetzten Transportprotokolle haben ihre Internet-Pendants beispielsweise mit dem Transmission Control Protocol (TCP) oder dem User Datagram Protocol (UDP) auf Schicht 4, während die Diagnoseprotokolle mit dem File Transfer Protocol (FTP) oder dem Simple Network Management Protocol (SNMP) auf der Anwendungsschicht (Schicht 7) vergleichbar sind. Jedes dieser Protokolle setzt auf der Schicht 3 auf, die für die Vermittlung der Daten im Netzwerk zuständig ist. Im Bereich der Internetprotokolle übernimmt diese Aufgabe das Internet Protocol (IP), für das bislang kein vergleichbares Protokoll im Bereich der automobilen Netzwerke existiert.

6.3.1 Internet Protocol (IP)

Das am häufigsten in Kommunikationsnetzwerken eingesetzte und verbreitete Internet Protocol trägt die Version 4 (IPv4) und wurde 1981 in der RFC 791 [RFC791] definiert. Mittlerweile existiert eine Version 6 dieses Protokolls (IPv6) [RFC2460], das in erster Linie der Adressknappheit von IPv4 mit derzeit 2^{32} möglichen Adressen auf 2^{128} mögliche Adressen begegnen soll. Für den Einsatz in in Fahrzeugen, die derzeit mit bis zu 80 Steuergeräten ausgestattet sein können, ist allerdings schon IPv4 mehr als ausreichend. Die bewährten Mechanismen, die IPv4 zum Datenrouting nutzt, lassen sich dennoch für Fahrzeug-Kommunikationsnetzwerke anpassen.

[32] Jeder Datenbus ist am Gateway-Steuergerät angeschlossen und auf direktem Weg zu erreichen. Die in aktuellen Fahrzeugen eingesetzten LIN-Busse werden als Subbusse nicht direkt adressiert, sondern Informationen über den LIN-Master weitergegeben.

Die Version 4 des Internet Protocols benutzt 32 Bit-Adressen, die in vier Blöcke von je einem Byte unterteilt werden und maximal 4.294.967.296 eindeutige Adressen ermöglichen. Eine IP-Adresse unterteilt sich in einen Netzwerkteil sowie einen Host-Adressteil, wobei sich die Teilung dynamisch durch die Angabe einer Subnet-Maske anpassen lässt. Zu einem IP-Netzwerk gehören die Teilnehmer, deren Netzwerkteil ihrer Adressen gleich ist. Dies ist die Voraussetzung für eine direkte Kommunikation der Teilnehmer untereinander. Bezogen auf das Fahrzeug-Netzwerk bedeutet ein gleicher Netzwerkteil, dass sich die Steuergeräte innerhalb einer Domäne befinden und direkt über ihren Datenbus miteinander kommunizieren können. Für alle anderen Fälle müssen Gateways die Daten zwischen den Domänen vermitteln.

Für die Umsetzung der (logischen) IP-Adressen zu physikalischen (Netzwerk)-Adressen kommen weitere Protokolle wie beispielsweise ARP (Adress Resolution Protocol [RFC826]) zum Einsatz. Derartige Protokolle sind im Kraftfahrzeug nicht notwendig und können über eine statische Zuordnung von Steuergeräten zu logischen Adressen seitens des Fahrzeugherstellers ersetzt werden. Die logischen Adressen werden jedoch für die Vermittlung der Daten im Fahrzeugnetzwerk benötigt. Hier setzt ein Routingprotokoll an. Die physikalischen Adressen entsprechen dagegen beispielsweise bei CAN-Netzwerken spezfischen CAN-Identifiern, die bestimmten Steuergeräten zugeordnet sind oder bei FlexRay bestimmten Slots, die für bestimmte Steuergeräte reserviert sind. Im Gegensatz zu den logischen Adressen, die fahrzeugweit eindeutig sein müssen, brauchen die physikalischen Adressen nur innerhalb ihrer Domäne eindeutig zu sein.

6.3.1.1 Aufbau des Internet-Protocols

Ein IP-Paket besteht aus einem Header und den dazu gehörigen Nutzdaten. Der Header enthält die protokollrelevanten Informationen eines IP-Pakets. In Abbildung 57 ist der Aufbau des IP-Headers dargestellt, der im Normalfall 20 Bytes lang ist und je nach verwendeten Optionen bis zu 60 Bytes groß werden kann. Die Informationen aus dem IP-Header, die für automobile Netzwerke interessant sind, sind in der Abbildung gekennzeichnet.

Die Version bezeichnet das eingesetzte IP-Protokoll (z.B. IPv4 oder IPv6), während der Header die Länge des IP-Headers in 32 Bit-Schritten angibt. Beide Informationen werden für ein Fahrzeug-Routingprotokoll nicht benötigt, da in dem Fall die verwendete Protokollversion für ein Fahrzeug spezifiziert ist. Das Type of Service-Feld kann zur Priorisierung von IP-Paketen eingesetzt werden (Quality of Service), was im Speziallfall der Diagnosekommunikation zwar interessant, aber unter Kosten/Nutzenaspekten wenig sinnvoll ist. Das Feld „Gesamtlänge“ gibt die Länge des gesamten Pakets inklusive Header an und kann bei IP maximal 65536 Bytes betragen. Für den Einsatz im Fahrzeug ist dieses Feld notwendig, wobei dort die maximale Blocklänge zur Minimierung der Headerlänge verringert werden kann. Die anschließenden 4 Bytes werden zur Steuerung von fragmentierten IP-Paketen benötigt. Beim Einsatz im Fahrzeug sind durch die relativen kurzen Paketlängen von acht Bytes ebenfalls Fragmentierungsmechanismen notwendig. Hier kann allerdings auf die bewährten Mechanismen der etablierten Transportprotokolle zurückgegriffen werden.

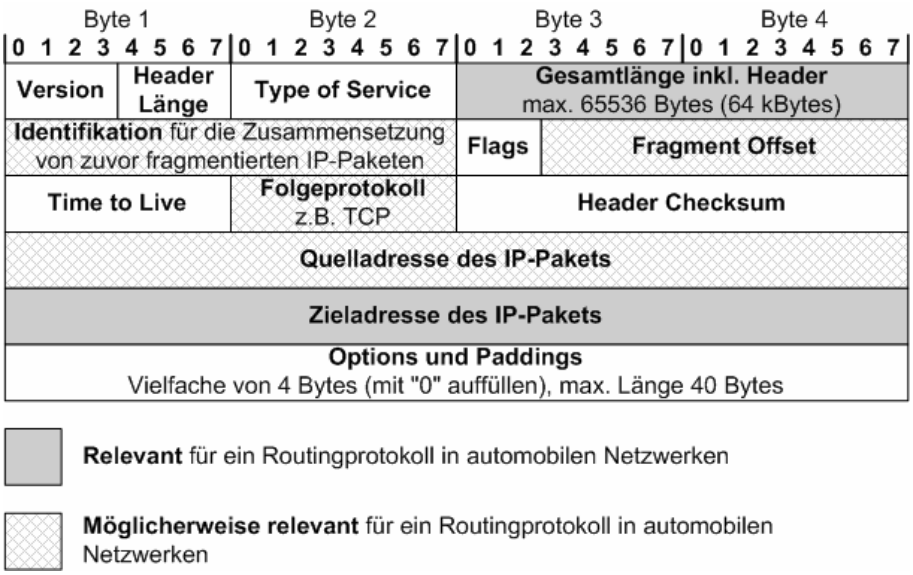

Abbildung 57: Aufbau des IP-Headers

Das Time-to-Live-Feld (TTL-Zeit) des IP-Headers wird im Internet benötigt, um Pakete bei fehlendem Ziel nicht endlos zu routen, sondern diese nach Ablauf der TTL-Zeit zu verwerfen, was für hochdynamische Netzwerke sinnvoll ist. Im eher statischen Fall eines automobilen Kommunikationsnetzwerks kann auf diese Information verzichtet werden. Im Protokollfeld ist das Folgeprotokoll kodiert, das sich im Nutzdatenteil des IP-Pakets befindet (z.B. für TCP-06h). Für ein Routingprotokoll im Fahrzeug wäre ein solches Vorgehen ebenfalls denkbar, in dem die bestehenden Protokolle (TP, KWP) vollständig von diesem gekapselt würden. In der Realisierung für das Labornetzwerk wurde allerdings darauf verzichtet, da solch ein Schritt einen zu großen Bruch mit bestehenden Prozessen bedeuten würde und die vollständige Entwicklung eines automobilen Routingprotokolls nicht im Vordergrund stand. Auf die Checksumme kann ebenfalls verzichtet werden, da die bestehenden automobilen Netzwerke und Protokolle durch ihre Konzeption die Übertragungssicherheit gewährleisten. Anschließend folgen noch die Quell- und die Zieladresse. Die Zieladresse ist essentiell, während durch den Verzicht auf die Quelladresse Bandbreite gespart wird. Da ein Routingprotokoll in der Regel nur zwischen Diagnosetester und Steuergerät eingesetzt wird, reicht für diesen Zweck eine Session-ID, die durch den Diagnosetester verwaltet wird.

Im Anschluss an den IP-Header folgen die Nutzdaten, die meist vollständige Protokolle kapseln. Das oft in Verbindung mit IP zum Einsatz kommende und im ISO/OSI-7-Schichtenmodell eine Schicht weiter oben angesiedelte TCP befindet sich beispielsweise ausschließlich im Nutzdatenbereich von IP. Würde man dieses Vorgehen auf die automobilen Netzwerke übertragen, bedeutet dies, dass die Transportprotokolle (TP 1.6, TP 2.0) und Diagnoseprotokolle (KWP, UDS) als solche weiter verwendet werden können. Die Einzelheiten, welche der IP-Mechanismen sich für ein automobiles Routingprotokoll eignen und wie diese angepasst wurden, beschreibt der nächste Abschnitt. Einen Schwerpunkt bei der Beschreibung bildet dabei die Umsetzung einer fahrzeugweit eindeutigen Adressierung.

6.3.2 Anpassung an automobile Anforderungen

Während man im Internet eine riesige Anzahl an unterschiedlichsten Netzteilnehmern in einer hochdynamischen Umgebung vorfindet, so stellen sich die Verhältnisse im Automobil vergleichsweise statisch dar. Die Anzahl der in einem Kraftfahrzeug verbauten Steuergeräte kann in extremen Fällen derzeit bis zu 100 betragen. Eine weitere Steigerung ist auf Grund des beschränkten Bauraums und dem andauernden Druck zur Kosten- und Gewichtsreduzierung nicht zu erwarten. Die Arten und Vernetzungsstrukturen der Steuergeräte stehen ebenfalls in engen Grenzen fest. Hier liegen die Unterschiede in den einzelnen Fahrzeugtypen und Ausstattungsvarianten.

Betrachtet man die Anforderungen zwischen automobilen Netzwerken und denen des Internets wird deutlich, dass das IP-Protokoll nicht direkt für den Einsatz im Kraftfahrzeug übertragbar ist. Zum einen kann auf einen großen Teil der Flexibilität von IP verzichtet werden, in dem vorhandenes apriori-Wissen eingesetzt wird. Zum anderen ist eine Beschränkung auf wenige Bytes notwendig, um den durch das Routingprotokoll verursachten Overhead möglichst gering zu halten. Die Anlehnung an IP ist dennoch sinnvoll, da davon ausgegangen wird, dass Fahrzeuge zukünftig mit dem Internet kommunizieren und dabei über eine eigene IP-Adresse erreichbar sein werden. Ein ähnlich logischer Aufbau des fahrzeuginternen IP-Protokolls bietet bei diesem Szenario den Vorteil der einfacheren Vermittlung zwischen den Protokollen. Darüber hinaus hat das IP-Protokoll in seinem Aufbau seine Leistungsfähigkeit im Internet bewiesen.

Der wesentliche Unterschied zwischen Internetprotokollen und automobilen Übertragungsprotokollen liegt in den verfügbaren Übertragungsraten und den daraus abgeleiteten maximal zur Verfügung stehenden Telegrammrahmen. Können diese bei den Internetprotokollen im Kilobyte-Bereich liegen, so wird im Fahrzeug alles an den acht Bytes eines CAN-Telegramms gemessen. Des Weiteren sind nahezu alle Internetprotokolle teilnehmerorientert, d.h. ein Netzteilnehmer kann direkt adressiert werden. Dem gegenüber steht der CAN-Bus als nachrichtenorientierter Datenbus, bei dem bei der Adressierung Anpassungen vorgenommen werden müssen.

6.3.3 Möglichkeiten zur Realisierung des Routing im Fahrzeug

Grundsätzlich bietet sich für einen flexiblen Architekturansatz eines Fahrzeugs die Verwendung eines Routingprotokolls sowohl für die Standard- als auch für die Diagnosekommunikation an. In diesem Fall wären dabei nicht nur Informationen über die Zieladressierung, sondern darüber hinaus auch über die Absenderadresse notwendig, um die Unabhängigkeit vom Übertragungskanal zu gewährleisten. Gleichzeitig würde dieser Schritt allerdings auch einen Verzicht auf die etablierten und bewährten Kommunikationsmechanismen bedeuten und den Overhead der Standardkommunikation erhöhen. Besser ist dagegen die Ausnutzung der bereits während der Festlegung der Systemspezifikation bekannten Informationen zur Sicherstellung der Kommunikation in heterogenen Netzwerken bei Einschränkungen hinsichtlich der Kommunikationsflexibilität. Unter diesen Aspekten reicht für einen ersten Schritt zur Einführung eines Routingprotokolls in das Fahrzeug die Beschränkung auf die Diagnosekommunikation. In dem Fall muss durch das

Protokoll sichergestellt werden, dass bei nicht vollständig bekanntem Aufbau der Fahrzeugarchitektur mit jedem einzelnen Steuergerät von außen kommuniziert werden kann. Wie so oft, existieren auch hier verschiedene Ansätze zur Lösung dieser Problemstellung, die im Folgenden kurz vorgestellt werden. Dabei muss zwischen dem eigentlichen Routingprotokoll (IP), das die Adressierung und Segmentierung festlegt, und den Vermittlungsmechanismen (RIP, Routing Information Protocol) unterschieden werden, die für die eigentliche Wegfindung zuständig sind.

6.3.3.1 Adaptives Routing

Die Vermittlung der Daten erfolgt anhand ihrer Zieladresse. Dafür muss ein Weg, die Route, zum Ziel bekannt sein, den die Daten im Netzwerk nehmen müssen und der aus so genannten Routingtabellen gelesen werden kann. Diese Routingtabellen können dynamisch aufgebaut und verändert werden oder aber stehen als statische Tabellen fest. Insbesondere in großen dynamischen Netzen wie beispielsweise dem Internet können die Routen starken Veränderungen unterliegen. Es kommen Netzwerksegmente hinzu oder können ausfallen. Daher existiert eine Vielzahl von Verfahren, wie diese Routen aktuell gehalten werden können. Sie benutzen dazu zwei grundlegende Verfahrensweisen. Die einzelnen Systeme teilen dem Netz als Broadcast mit, wer ihre Nachbarn sind und nach einiger Zeit ist die vollständige Topologie des Netzes bekannt. Der andere Ansatz besteht darin, dass die Systeme nur ihren direkten Nachbarn Informationen darüber geben, wie das Netz für sie aussieht. Die Vor- und Nachteile sowie Details zu den dahinter stehenden Konzepten werden hier nicht weiter erläutert.

Im Fall des abgeschlossenen Systems „Kraftfahrzeug“ kann die Netzwerktopologie als feststehend und bekannt vorausgesetzt werden. Ein dynamisches Routing ähnlich dem des Internets ist somit nicht erforderlich. Allerdings wäre in einigen Fällen eine begrenzte Flexibilität bei den Routingtabellen sinnvoll. Fahrzeuge können abhängig von ihrer Ausstattung unterschiedliche Netzwerktopologien besitzen und müssten daher mit speziell angepassten Routingtabellen ausgerüstet werden. Hier bietet sich an, dass sich Steuergeräte im Fahrzeug durch eine per Diagnosesitzung angestoßene Anpassung autonom ihre Routingtabellen aufbauen. Noch größere Flexibilität erreicht man, indem diese Routinginformation bei jedem Systemstart aktualisiert werden, was allerdings nur unter sehr strengen Zeitrestriktionen geschehen kann. Diese Überlegungen führen in die Richtung einer automobilen Middleware, wie sie im Rahmen von AUTOSAR entwickelt wird. Für den Austausch von Routing-informationen bietet dabei sich das Netzwerkmanagement an, welches in Fahrzeugen beispielsweise für das „Aufwachen“ oder zum „Schlafenlegen“ von Steuergeräten genutzt wird.

6.3.3.2 Statisches Routing

Im Gegensatz zum adaptiven Routing werden beim statischen Routing die Routing-tabellen fest im Speicher des jeweiligen Steuergeräts abgelegt. Bei jeder Änderung der Netzwerktopologie müssen diese Tabellen entsprechend angepasst werden, was dieses Verfahren sehr unflexibel macht. Der Vorteil liegt in der einfacheren Implementierung und dem Verzicht auf zusätzliche Mechanismen zur Wegfindung

und damit einen geringen Overhead. In heutigen Fahrzeugen findet man zum überwiegenden Teil statisches Routing im Gateway-Steuergerät vor. Es besitzt eine Verbindungsmatrix, anhand derer die Daten von einem Datenbus auf einen anderen geroutet werden.

Für das Labornetzwerk wird ebenfalls ein statisches Routing verwendet, da die Netztopologie bekannt ist und das adaptive Routing nicht weiter betrachtet wird. Dazu ist in jedem Gateway eine Tabelle mit einem Eintrag für jeden möglichen Zielknoten angelegt. Jeder Eintrag besitzt eine Spalte, auf welchem Datenbus der Zielknoten lokalisiert und über welches Businterface des Gateways dieser Knoten erreichbar ist. Das Gateway analysiert die Zieladresse der eintreffenden Daten mit Hilfe dieser Routingtabelle und kann dann entscheiden, ob und auf welcher Schnittstelle die Daten weitergeleitet oder aber verworfen werden.

6.3.4 Ansätze eines automobilen Routingprotokolls

Ist die Netzwerktopologie den einzelnen Gateways bekannt, wird ein Routingprotokoll benötigt, das es erlaubt, die Zieladressinformationen zusammen mit den Daten zu übertragen. Gleichzeitig muss solch ein Routingprotokoll dafür sorgen, dass die maximal erlaubte Nutzdatengröße eines Telegramms (PDU, engl. Protocol Data Unit) nicht überschritten wird. Dazu werden größere Datenmengen beim Absender segmentiert und am Ziel wieder zusammengesetzt. Für die Realisierung eines Routingprotokolls wurden mehrere Ansätze entwickelt, die im Folgenden kurz vorgestellt werden.

6.3.4.1 Transparentes Routingprotokoll

Der generalistische Ansatz eines Routingprotokolls und gleichzeitig die sauberste Lösung ist die Realisierung eines transparenten Routingprotokolls auf der Schicht 3 des ISO/OSI-7-Schichtenmodells. Ähnlich dem IP könnte solch ein Protokoll als AutomotiveIP bezeichnet werden. Es kapselt die Protokolle der höheren Schichten und kümmert sich um eine geeignete Adressierung und Segmentierung der Datenpakete. Allerdings würde die Einführung der Routingschicht vor allem bei Datenbussen mit relativ kleinen Nutzdatengrößen wie CAN und LIN den Kommunikationsoverhead signifikant erhöhen. Insbesondere die Segmentierung der Datenpakete muss für eine effiziente Kommunikation an den jeweiligen Übertragungskanal angepasst werden. Damit würde ein Großteil der Kommunikationsaufgaben, die in heutigen Kraftfahrzeugen durch die Transportprotokolle realisiert sind, in das AutomotiveIP übergehen. Über die Transportprotokolle müssten bei Bedarf die Datenintegrität sichergestellt werden, für die das Routingprotokoll nicht verantwortlich ist.

Die Lösung des transparenten Routings auf Basis eines AutomotiveIP bietet den Vorteil der strikten Trennung zwischen Transport- und Vermittlungsschicht bei gleichzeitiger Unabhängigkeit vom Übertragungsmedium. Für zukünftige Fahrzeug-Architekturen ist dieser Ansatz sehr wahrscheinlich der langfristig geeignete, besonders bei einem zu erwartenden Einsatz von Ethernet im Fahrzeug. Für die Umsetzung im Labornetzwerk allerdings bedeutet die Implementierung eines solchen AutomotiveIP einen zu radikalen Bruch mit den bestehenden Transportprotokollen.

Insbesondere der Prozess der Update-Programmierung im Fahrzeug ist sehr eng mit den bestehenden Transportprotokollen verzahnt. Ein vollständig neues Protokollkonzept würde keine Aussagen über die Machbarkeit der Kommunikation in derzeit vorhandenen heterogenen Kommunikationsstrukturen erlauben. Darüber hinaus gehören zu einem AutomotiveIP entsprechende Vermittlungstabellen, die zumindest teilweise dynamisch aufgebaut werden müssen, um die Flexibilität zu erhalten. Derartige Mechanismen, wie sie beispielsweise zu einer (automobilen) Middleware gehören, werden im Rahmen dieser Arbeit nicht weiter vorgestellt. Für das Labornetzwerk reichen statische, fest programmierte, Vermittlungstabellen aus. Für den Einsatz in einem realen Fahrzeugnetzwerk wären solche statischen Tabellen nicht geeignet. Beispielsweise müssten diese für jedes Fahrzeug speziell auf die jeweiligen dort verbauten Systeme abgestimmt sein.

6.3.4.2 Angepasstes Routingprotokoll

Der Ansatz, wie er in der vorliegenden Arbeit umgesetzt und als „angepasstes“ Routing bezeichnet wird, ist nicht mehr unabhängig vom Übertragungssystem. Während die Adressinformation netzwerkweit eindeutig und unabhängig vom Übertragungssystem ist, wird die Segmentierung an das jeweils zu Grunde liegende Übertragungsverfahren angepasst. Dazu wird die Tatsache genutzt, dass die einzelnen Verbindungsknoten zwischen den Netzsegmeneten als Gateways ausgeführt sind. Diese arbeiten auf Schicht 7 des ISO/OSI-7-Schichtenmodells und können die Daten auf Applikationsebene behandeln (siehe auch Abschnitt 3.5.1.4).

Der Teil des Routings für die Adressierung kann somit in den Bereich der Applikation verlagert werden. Für die Segmentierung dagegen können auf dem CAN-Bus das Transportprotokoll TP 2.0 benutzt werden. In dem Fall ist zusätzlich eine Adressumsetzung zwischen netzwerkweit eindeutiger Zieladressierung und der lokalen Adressierung, wie sie TP 2.0 nutzt, notwendig. Für die anderen Übertragungssysteme wie Bluetooth und FlexRay existieren noch keine automobilen Transportprotokolle. Hier wurden entsprechend angepasste Versionen des CAN-Routingprotokolls entwickelt, um eine möglichst einfache Umsetzung der einzelnen Protokolle beim Routing zu ermöglichen. Die Einzelheiten zum Telegrammaufbau beschreiben der Abschnitte 6.4.2 bis 6.4.4.

6.3.4.3 Sitzungsbasiertes Routingprotokoll

Ein weiterer Ansatz für ein Routingprotokoll ist die Einführung einer Routingsession. Ähnlich zur Diagnosesession würde über eine Eröffnungsnachricht auf einem Übertragungssystem die Routinginformationen wie Datenlänge, Absender- und Zieladresse ausgetauscht. Die Kommunikationspartner bestätigen sich die Informationen und handeln eine Routingsession-ID aus, die im weiteren Verlauf der Kommunikation den Daten vorangestellt wird. Ist ein Kommunikationspartner noch nicht der Empfänger der Daten, wird eine weitere Session auf dem nächsten Übertragungssystem initiiert. Diese Art des Routing besitzt den Vorteil, dass nur marginale Änderungen an bestehenden Protokollen vorgenommen werden müssen. Für das Labornetzwerk bietet das angepasste Routing jedoch den gleichen Nutzen bei weniger Implementierungsaufwand als das sitzungsbasierte Routingprotokoll. Daher wurde im Labornetzwerk das angepasste Routingprotokoll realisiert.

6.3.5 Adressierung für ein heterogenes Kommunikationsnetzwerk

Zur Realisierung eines Routingprotokolls muss eine fahrzeugweit eindeutige Adressierung festgelegt werden. Diese sollte auch für zukünftige Kommunikationsnetzwerke erweiterbar ausgelegt sein und trotz allem mit möglichst wenig Datenbits auskommen. Dazu sind einige Vorüberlegungen und Einschränkungen bei den Freiheitsgraden erforderlich.

Für die Auslegung des Routingprotokolls kann von der minimalen Telegrammlänge von acht Bytes des CAN-Protokolls ausgegangen werden. Unter dieser Annahme stehen bei Berücksichtigung einer möglichst hohen Protokolleffizienz maximal zwei Bytes pro Telegramm für ein Routingprotokoll zur Verfügung und bedeutet für das Routing die Beschränkung auf 25 % eines CAN-Telegramms. Die zwei Bytes (16 Bit), die für das Routing zur Verfügung stehen, müssen entsprechend effektiv genutzt werden. Dazu wurde der logischen Adressierung der Steuergeräte der meiste Platz eingeräumt, um in der Adresse, ähnlich zu einem IP-Paket, die Netzwerkstruktur mit abzubilden. Die Aufteilung der zur Verfügung stehenden 16 Bit zeigt Abbildung 58.

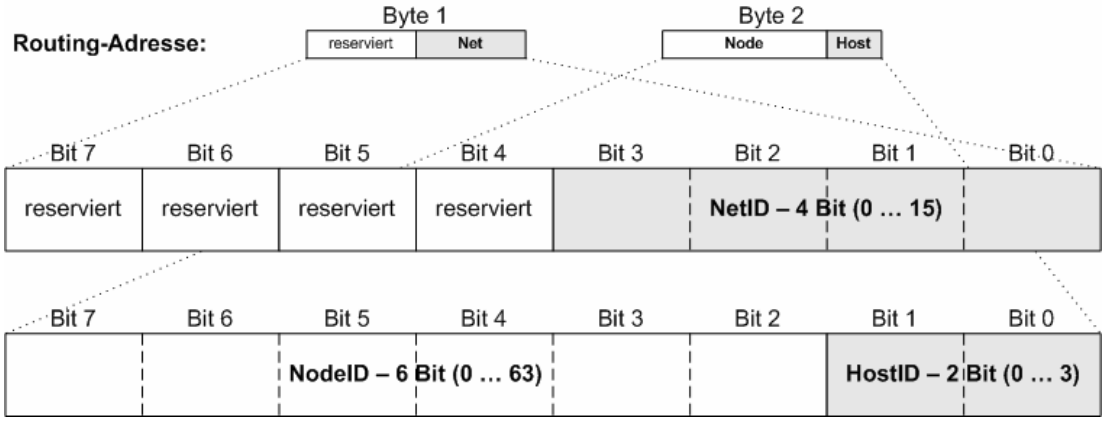

Abbildung 58: Aufbau der Adressbytes für das Routing

Für die Adressierung eines Netzwerksegments (NetID), gleichbedeutend mit einer Domäne wie beispielsweise dem Antriebsstrang, werden vier Bits verwendet. Damit lassen sich sich 15 verschiedene Domänen (1-15) adressieren. Die verbleibende Möglichkeit mit der Netzadresse Null kann für eine Sonderadressierung wie zum Beispiel für einen Netzwerk-Broadcast genutzt werden. Kommunikationsnetzwerke in aktuellen Fahrzeugen sind derzeit bis in sechs verschiedene Domänen unterteilt, so dass eine Domänenadressierung mit vier Bits auch für zukünftige Fahrzeugnetze ausreichend dimensioniert ist.

In den Domänen befinden sich die Steuergeräte, die an die jeweiligen Datenbussen angeschlossen sind. Jedes Datenbussystem kann eine maximale Anzahl von Teilnehmern bedienen. Beim LIN-Bus können bis zu 16 Teilnehmer angeschlossen sein, während FlexRay bis zu 64 Teilnehmer bedienen kann. Einzig am CAN-Bus könnten theoretisch so viele Teilnehmer wie Nachrichten-Identifier vorhanden sein. Allerdings ist die Teilnehmeranzahl beim CAN-Bus durch die verwendeten Treiberbausteine auf 32 bis 64 Teilnehmer beschränkt. Es kann aber davon ausgegangen

werden, dass die Fahrzeug-Bussysteme nicht bis zu ihrer Maximalausbaustufe verbaut werden. Mit der sechs Bit langen Knotenadresse (NodeID) lassen sich 64 Teilnehmer pro Domäne adressieren. Auch hier bietet sich ähnlich zu IP und der NetID die Einführung einer speziellen Adresse, beispielsweise Null, an, die als Broadcast alle Teilnehmer der Domäne adressiert.

Für den zu erwartenden Anstieg an Funktionen in zukünftigen Fahrzeugen mit immer leistungsfähigeren Steuergeräten ist davon auszugehen, dass eine Applikation ein Steuergerät nicht mehr exklusiv für sich allein nutzen kann. Vielmehr werden Applikationen parallel auf einem Steuergerät ablaufen. Aus diesem Grund ist in der Adressierung eine zwei Bit große HostID vorgesehen, mit der vier verschiedene Applikationen eines Steuergeräts angesprochen werden können. Bei Bedarf kann das Adressformat entsprechend verschlankt werden, indem auf überflüssige Adressteile verzichtet wird. Beispielsweise kann beim Routing auf den LIN-Bus die NetID entfallen, da es unwahrscheinlich ist, dass es hierarchisch unterhalb des LIN-Busses einen weiteren Subbus geben wird.

Bei der vorgestellten Adressierung handelt es sich um Minimalanforderungen, die aus den voran gegangenen Überlegungen an zukünftige Kraftfahrzeugnetzwerke abgeleitet wurden. Die NetID (4 Bit), die NodeID (6 Bit) und die HostID (2 Bit) benötigen zusammen 12 Bit, so dass von den 16 zur Verfügung stehenden Bits vier für weitere zukünftige Anforderungen genutzt werden können. Nutzt man beispielsweise bei der HostID eine Adresse als Broadcastadresse und stellen sich die verbleibenden drei Host-Adressen als ungenügend heraus, könnte die HostID leicht auf drei Bits erweitert werden. Damit lassen sich sieben Applikationen inklusive Broadcast adressieren. Für die Belange des Labornetzwerks sind die vorgestellten Dimensionierungen der Adressen ausreichend und wurden so verwendet.

6.3.6 Besonderheiten bei der Adressierung der Diagnoseschnittstelle

Bei der Umsetzung des Routingprotokolls wurde zur Verringerung des Overheads auf die Angabe der Absenderadresse verzichtet. Da das Protokoll nur für den Einsatz von Diagnosenachrichten ausgelegt ist, ist der Verzicht auf den Absender unproblematisch. Jede Diagnosenachricht, die ein Steuergerät empfängt, hat einen Diagnosetester (Diagnoseclient) als Ursprung. Bei der Festlegung, dass es nur eine einzige aktive Diagnosekommunikation von einem Diagnoseclient mit einem Steuergerät geben darf, würde dieses Vorgehen ausreichen. Allerdings verspricht gerade die parallele Kommunikation über den breitbandigen Backbone bei der Update-Programmierung einen sehr großen Nutzen. Aus diesem Grund müssen weitere Vereinbarungen getroffen werden.

6.3.6.1 Bit-Signalisierung der Diagnoseadresse

Ein Ansatz, Diagnosenachrichten für den Diagnoseclient zu kennzeichnen, besteht in der Verwendung eines der reservierten Bits der zwei Byte Zieladresse. Die Routingmechanismen in den Gateways überprüfen dieses Diagnose-Bit. Ist das Diagnose-Bit nicht gesetzt, wird die Zieladresse wie bisher ausgewertet und das Telegramm zum Steuergerät geschickt. Im anderen Fall wird die Nachricht zur Diagnoseschnittstelle geroutet. In der ursprünglichen Zieladresse, die von den

Gateways dann nicht weiter beachtet wird, steht immer noch das Steuergerät und ist damit zur Absenderadresse umfunktioniert. Anhand dieser Adresse kann ein Diagnoseclient die eintreffenden Daten den jeweiligen Steuergeräten zuordnen.

Dieser Weg zur Signalisierung von Diagnosenachrichten hat allerdings den Nachteil, dass für ein Steuergerät nur eine aktive Diagnosesitzung erlaubt ist. Bei parallelen Sitzungen für ein Steuergerät kann der Diagnoseserver (die Diagnoseschnittstelle) keine Zuordnung zwischen verschiedenen Diagnoseclients treffen. Daher wurde für das Labornetzwerk der allgemeinere Ansatz über eine SessionID umgesetzt, der im folgenden Abschnitt vorgestellt wird.

6.3.6.2 Verwendung einer SessionID für die Diagnoseadressierung

Bei jedem Aufbau einer Diagnosesitzung mit einem Steuergerät generiert der Diagnoseclient eine eindeutige, ein Byte große, SessionID, die im Eröffnungstelegramm dem Steuergerät mitgeteilt wird. Für die Kommunikation vom Steuergerät zum Diagnoseclient wird über die NetID der Diagnoseserver (Diagnoseschnittstelle) adressiert. In dem Adressbyte, welches die NodeID und die HostID enthält, wird nun für Nachrichten zum Diagnoseclient die aus dem Eröffnungstelegramm bekannte SessionID übermittelt. Anhand dieser SessionID kann der Diagnoseserver entscheiden, welcher Diagnoseclient diese Verbindung initiiert hat und die Telegramme dorthin übertragen. Bei parallelen Diagnosesitzungen eines Diagnoseclients mit einem Steuergerät kann der Diagnoseclient mit Hilfe der SessionID ebenfalls die Kommunikationen richtig zuordnen.

Da der Ansatz der SessionID etwas flexibler als die Bit-Signalisierung aus Abschnitt 6.3.6.1 ist, wurde er im Labornetzwerk realisiert. Die möglichen, vom Ansatz her unterstützten, parallelen Sitzungen mehrerer Diagnoseclients sind bei der Implementierung im Labornetzwerk unberücksichtigt geblieben, da dafür kein Bedarf bestand. Allerdings ist eine Integration sehr leicht möglich. Beispielsweise wäre beim Kommunikationsaufbau zwischen Diagnoseclient mit dem Diagnoseserver denkbar, dass der Diagnoseserver dem Client einen Bereich von zu nutzenden SessionIDs mitteilt. Damit wäre die Eindeutigkeit der im Netzwerk verwendeten SessionIDs auch Diagnoseclient-übergreifend sichergestellt.

6.4 Umsetzung der Kommunikation im Labornetzwerk

Die Überlegungen bei der Realisierung eines Routingsprotokolls für das Labornetzwerk führten zu einem Ansatz, der die einzelnen Datenbusse unterschiedlich behandelt. Ein generalistischer Ansatz, der transparent für alle Datenbusse, unabhängig von Telegrammlängen und Adressierungsarten funktioniert, erscheint unter den zu erwartenden Nutzen als zu aufwendig. Gerade bei kleinen Telegrammlängen wie beim CAN- oder LIN-Bus würde ein Großteil der verfügbaren Datenübertragungsrate für Verwaltungsinformationen verschwendet, so dass die Effizienz des Kommunikationssystems leidet. Ein Ziel für diese Arbeit ist aber gerade eine Erhöhung der Effizienz, die eine Übertragung großer Datenmengen in das Fahrzeug in kurzer Zeit ermöglich soll. Aus diesem Grund werden nacheinander die verschiedenen Umsetzungen des Routings für die einzelnen Teilnetzwerke (Diagnoseschnittstelle, Backbone, CAN-Bus) in den folgenden Abschnitten

vorgestellt, wie sie im Labornetzwerk realisiert wurden. Die Nutzdaten der einzelnen Telegramme wurden für die Übertragung in einzelne Pakete von je acht Bytes unterteilt, um eine einfache Konvertierung auf das CAN-Nachrichtformat zu ermöglichen.

6.4.1 Adressfestlegungen im Labornetzwerk

Die wesentliche Voraussetzung für das Routingprotokoll besteht dabei in der Vergabe von fahrzeugweit eindeutigen Netzwerkadressen für die angeschlossenen Steuergeräte. Ausgehend von der Beschreibung des Labornetzwerks aus Kapitel 5 lassen sich vier verschiedene Domänen innerhalb des Labornetzwerks identifizieren. Diese wurden unter Beachtung der Broadcast-Adresse ‚0' nacheinander durch nummeriert. Der FlexRay-Backbone erhält die Netzadresse ‚1', während die beiden CAN-Busse unter den Netzadressen ‚2' und ‚3' erreichbar sind. Der LIN-Bus bekommt die Netzadresse ‚4'. Die Steuergeräte der einzelnen Bussysteme werden ebenfalls mit einer NodeID, beginnend bei eins, durch nummeriert. Auf allen Steuergeräten des Labornetzwerks läuft nur eine einzige Applikation, so dass die HostID überall die Adresse ‚1' besitzt. Eine Ausnahme stellt der Diagnose-Client mit der Netzadresse ‚15' dar. Hier wird nicht direkt ein Netzwerk, sondern die Diagnoseschnittstelle adressiert. Einen Überblick über die Adressen, die in dem Labornetzwerk vergeben wurden, gibt die Abbildung 59.

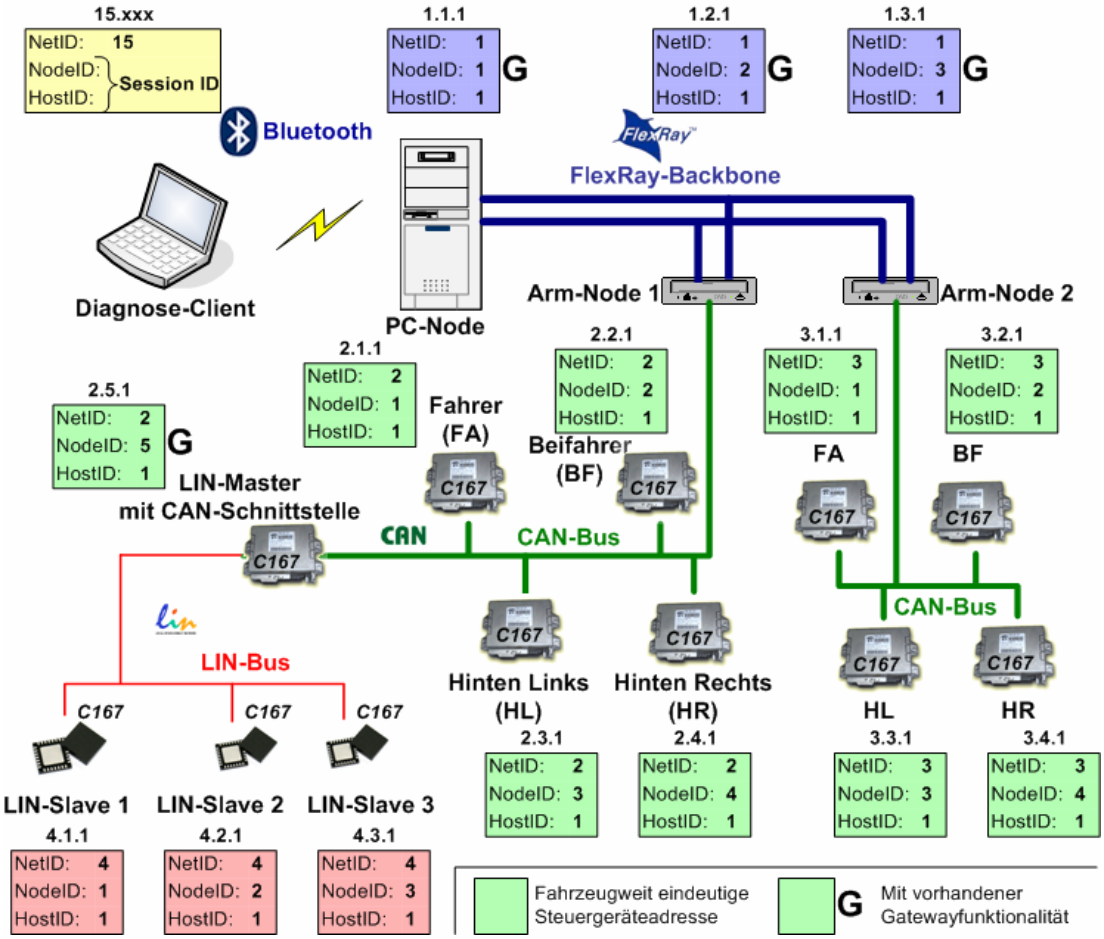

Abbildung 59: Adressfestlegungen im Labornetzwerk

6.4.2 Kommunikation des drahtlosen Diagnosekanals

Die Diagnoseschnittstelle des Labornetzwerks wurde mit Bluetooth unter Linux umgesetzt. Linux nutzt den BlueZ-Stack, der die Bluetooth-Funktionen kapselt und eine einheitliche Socket-Schnittstelle (API) für die Anwendungen zur Verfügung stellt. Die Realisierung mittels Sockets erlaubt eine maximale Telegrammlänge von 2048 Byte pro Sende- oder Empfangsanweisung. Um das Funkmedium nicht mit einer einzigen Diagnosesitzung voll auszulasten und anderen Bluetooth-Teilnehmern ebenfalls Zugriff auf den Diagnoseserver zu ermöglichen, wurde die Realisierung der Diagnosekommunikation auf eine maximale Nutzdatenlänge von 1024 Byte pro einzelnem Telegramm beschränkt. Jedes dieser Pakete enthält einen acht Byte großen Header, in dem neben der Art der Daten die Routing- und Adressinformationen enthalten sind. Abhängig von der Telegrammart können zusätzlich zum Header noch weitere Daten (bis zu 1016 Byte) in einem Telegramm übertragen werden. Bei der Update-Programmierung ist es unter Umständen notwendig, mehrere Megabytes vom Diagnosetester in das Fahrzeug zu transferieren. Daher unterstützt das eingesetzte Nachrichtenformat eine Segmentierung der Daten, mit der eine maximale Datengröße von 63,5 MByte in einer Session übertragen werden kann.

6.4.2.1 Telegramme vom Diagnose-Client zum Diagnose-Server

Bei der Kommunikation des Diagnose-Clients mit dem Diagnose-Server werden drei verschiedene Telegrammstrukturen benötigt, die in Abbildung 60 dargestellt sind. Jedes Telegramm besteht aus einem Header, der mit acht Byte ausreichend lang ist. Je nach Kommando können sich daran Daten bis zu einer Länge von 1016 Byte anschließen. Allen Telegrammen gemeinsam sind die ersten drei Bytes, die eine eindeutige SessionID der Diagnosesitzung und die Zieladresse des Steuergeräts enthalten. Das vierte Byte entscheidet über die Art des Telegramms.

Da die Diagnosekommunikation allein vom Diagnose-Client aktiv gesteuert wird, während der Diagnose-Server, das Fahrzeugnetzwerk, passiv agiert, werden die überwiegenden Nachrichten Kommandos enthalten. Im vierten Byte ist deshalb der Bereich von 00h-7Fh für Kommandos reserviert (z.B. 01h für ‚Status abfragen'). Der Diagnose-Server leitet diese Kommandos zum jeweiligen Steuergerät, das darauf bei Bedarf eine positive Bestätigung mit eventuellen Antwortdaten oder eine negative Antwort liefert. Die restlichen vier Bytes des Kommando-Telegramms sind derzeit nicht belegt und können für spätere Erweiterungen Kommando-spezifisch genutzt werden.

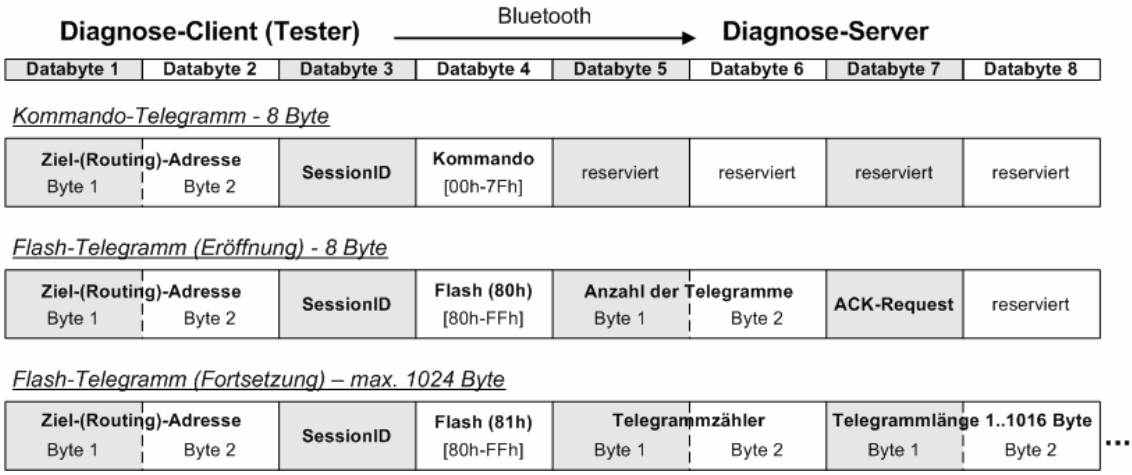

Abbildung 60: Aufbau der Telegramm(header) zwischen Diagnoseclient und Diagnoseserver

Einen Sonderfall stellt die Update-Programmierung dar, die deshalb ein spezielles Telegrammformat besitzt. Die Absicht zu Flashen wird ebenfalls als Kommando-Telegramm mit dem Kommando ‚80h' verschickt. Die dem Kommando-Byte folgenden zwei Bytes enthalten die Anzahl der Flash-Telegramme, die diesem Telegramm folgen. Zusätzlich enthält der Telegrammheader einen expliziten Acknowledge-Request. Dieser signalisiert dem Steuergerät, dass nach einer festgelegten Anzahl von Telegrammen[33] ein Acknowledge gesendet werden muss.

Die zu programmierenden Daten werden anschließend in Telegrammen verschickt, die einschließlich Header bis zu 1024 Byte lang sein können. Über den Telegrammzähler kann der Server kontrollieren, ob Telegramme verloren gegangen sind. Die Information über die Telegrammlänge sagt aus, wie viele der im Telegramm enthaltenen Datenbytes noch relevant sind, da zumindest das letzte Telegramm nicht die volle Telegrammlänge von 1024 Byte besitzen muss.

6.4.2.2 Telegramme vom Diagnose-Server zum Diagnose-Client

Der Diagnose-Server leitet Bestätigungs- und Datentelegramme aus dem Netzwerk zum Diagnose-Client (Abbildung 61). Die Daten aus dem Fahrzeug dürfen dabei eine Länge von 1016 Byte nicht überschreiten, da zumindest für das Labornetzwerk keine weitere Segmentierung in diese Kommunikationsrichtung vorgesehen wurde. Über die SessionID kann der Diagnose-Client die Daten einer Kommunikation zuordnen. Das dritte Byte enthält die Information, dass es sich um ein Acknowledge-Telegramm handelt.

[33] Dieses Vorgehen ist an das TP 2.0 vom CAN-Bus angelehnt, wo nach einer bestimmten Anzahl von Telegrammen, meist fünfzehn, ein Acknowledge erwartet wird. Da hier ein heterogenes Kommunikationsnetzwerk mit unterschiedlichen Telegrammlängen vorliegt, kann zwar die Anzahl der bestätigten Telegramme für jedes Kommunikationssystem gleich sein, besteht aber aus einer unterschiedlichen Menge an Datenbytes.

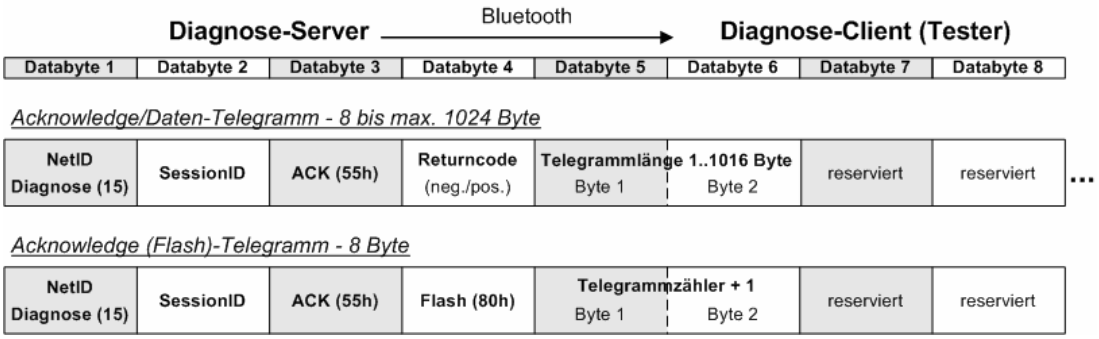

Abbildung 61: Aufbau der Telegramm(header) zwischen Diagnoseserver und Diagnoseclient

Antworten auf allgemeine Kommando-Telegramme enthalten im vierten Byte einen Returncode, der eine Aussage über die positive oder negative Bearbeitung des Kommandos trifft. In den folgenden zwei Bytes ist die Länge der eventuell im Telegramm enthaltenen Daten codiert, die sich an den acht Byte langen Header anschließen. Handelt es sich bei dem Returncode um den Wert ‚80h', bedeutet dies den Sonderfall eines Bestätigungstelegramm auf einen Programmierungvorgang. Die darauf folgenden zwei Byte enthalten statt der Telegrammlänge den Telegrammzähler, der als nächstes vom Diagnose-Server erwartet wird.

6.4.3 Kommunikation auf dem FlexRay-Backbone

Der Diagnose-Server ist an den FlexRay-Backbone angeschlossen und leitet vom Diagnose-Client eintreffende Nachrichten weiter. Der grundsätzliche Aufbau der Telegramme bleibt dabei bestehen. Ausgehend von einem acht Byte großem Header können die Telegramme allerdings die maximal möglichen 32 Byte der FlexRay-Telegramme im Labornetzwerk verwenden[34].

6.4.3.1 Telegramme vom Diagnose-Server zum Gateway-Steuergerät

Die Kommando-Telegramme des Diagnose-Clients mit ihrer Länge von acht Byte können auf dem FlexRay direkt weiter geleitet werden (Abbildung 62). Auf Grund der Beschränkung der maximalen Telegrammlänge von 32 Byte müssen die 1024 Byte langen Telegramme mit Flashdaten dagegen weiter segmentiert werden. Für die Anzahl der maximalen Telegramme sind zwei Byte nicht mehr ausreichend und benötigen ein weiteres Byte. Dagegen wird bei der Codierung der Telegrammlänge nur noch ein Byte benutzt, so dass nur geringfügige Änderungen an der Headerstruktur nötig sind.

[34] Der im Labornetzwerk eingesetzte FlexRay-Datenbus kann durch sein Prototypenstadium nur maximal 32 Byte große Telegramme benutzen. Nach Spezifikation wären auch Telegramme im dynamischen Teil, der hier zur Übertragung der Flashdaten genutzt wird, mit einer Länge bis zu 254 Byte möglich.

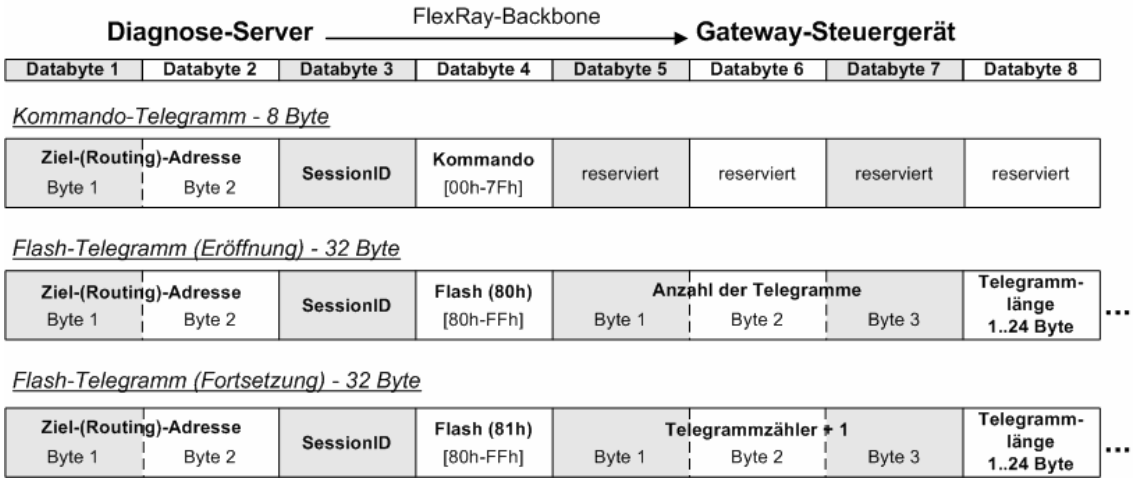

Abbildung 62: Aufbau der Telegramm(header) zwischen Diagnoseserver und FlexRay-Gateway

6.4.3.2 Telegramme vom Gateway-Steuergerät zum Diagnose-Server

Datentelegramme, die ein Gateway von einem Steuergerät zum Diagnose-Server schickt, benötigen eine Segmentierung, wie sie in Abbildung 63 dargestellt ist. Der Aufbau ist ansonsten der gleiche, wie bei der Bluetooth-Übertragung. Eine Änderung bezüglich der Bestätigungstelegramme für die Update-Programmierung wurde auf Grund der Eigenschaften des FlexRays vorgenommen. Es wird davon ausgegangen, dass der als Sicherheitsbus ausgelegte FlexRay keine Telegramme verliert. Daher werden keine Informationen über den Telegrammzähler mitgeschickt.

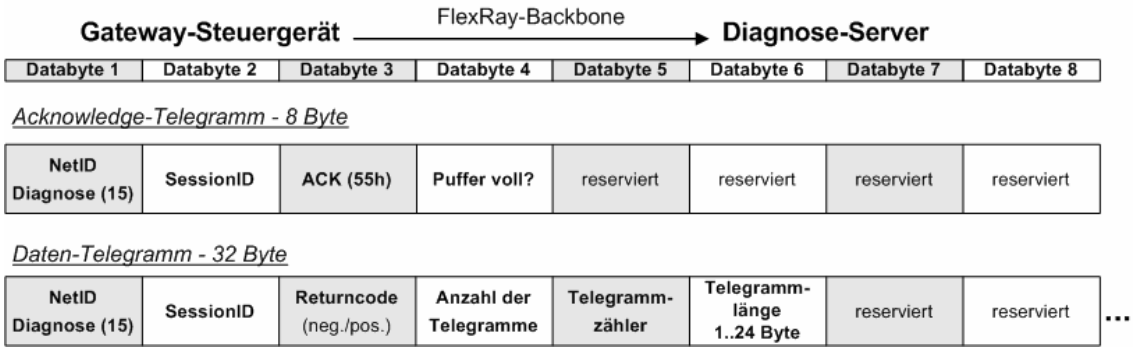

Abbildung 63: Aufbau der Telegramm(header) zwischen FlexRay-Gateway und Diagnoseserver

Die Daten können auf dem FlexRay mit mehreren Megabit pro Sekunde zum Gateway übertragen werden, während das Gateway die Daten über einen 500 kBit/s CAN-Bus zu einem Steuergerät schicken muss, dass diese Daten empfängt und programmiert. Daher hat das Gateway-Steuergerät im vierten Byte des Acknowledge-Telegramms die Möglichkeit, die weitere Übertragung von Flashdaten zu stoppen, falls die Datenpuffer vollgelaufen sind. Die Fortsetzung der Übertragung wird ebenfalls darüber signalisiert.

6.4.4 Kommunikation auf den CAN-/LIN-Datenbussen

Auf eine Darstellung der Telegramme des CAN-Datenbusses wird verzichtet, da sich hier keinerlei Änderungen zu den obigen acht Byte-Telegrammen ergeben. Diese werden in der gleichen Form vom Gateway auf den CAN-Bus unter einer festgelegten CAN-ID weitergeleitet. Für Telegramme, die Daten enthalten und länger als acht Byte sind, wird das bewährte Transportprotokoll TP 2.0 eingesetzt. Die Zieladresse, wie sie für das Routing vereinbart wurde, wird dabei den Daten, die über einen TP 2.0-Kanal gesendet werden, vorangestellt. Bei der Übertragung von Flashdaten werden somit acht Byte für den Telegrammheeader benötigt und die übrigen 4088 Bytes eines TP-Kanals sind vollständig für die Nutzdaten verwendbar.

Beim LIN-Datenbus ergeben sich ebenfalls keinerlei Änderungen zur derzeit im Fahrzeug verwendeten Kommunikation. Alle Telegramme, die eine Länge bis zu acht Byte besitzen, können direkt auf den LIN-Bus versendet werden. Für die Übertragung von Flashdaten ist ein entsprechendes Transportprotokoll, das sehr ähnlich zum TP 2.0 sein wird, zu verwenden. Die theoretischen Vorüberlegungen zeigen die prinzipielle Machbarkeit eines solchen Vorgehens. Eine Realisierung solch eines Transportprotokolls für den LIN-Datenbus wurde im Rahmen der Arbeit nicht umgesetzt.

6.5 Beispielhafte Kommunikation im Labornetzwerk

Anhand einer beispielhaften Update-Programmierung eines der CAN-Steuergeräte im Labornetzwerk, sollen die in den vorherigen Abschnitten beschriebenen Mechanismen zur Kommunikation in heterogenen Fahrzeugnetzwerken veranschaulicht werden. Dazu wird das CAN-Steuergerät des Beifahrer-Fensterhebers mit der Netzwerkadresse ‚2.3.1' neu programmiert.

6.5.1 Standardkommunikation

Das Labornetzwerk ist in einem normalen Betriebszustand und die Applikationen der einzelnen Steuergeräte sind funktionsfähig. Über den statischen Teil des Kommunikationszyklus vom FlexRay-Backbone wird die aufgezeichnete Antriebsstrang-Standardkommunikation eines PQ 24 getunnelt (siehe Abschnitt 5.2.2.4). Die CAN-Steuergeräte arbeiten ihre fest eingespeicherten Applikationen innerhalb ihrer Netzwerke ab. Es findet somit keinerlei domänenübergreifende Kommunikation statt.

6.5.2 Aufbau einer Diagnosesitzung

Über einen Laptop mit Bluetooth-Schnittstelle, der als Diagnose-Client fungiert, werden verfügbare Bluetooth-Geräte gesucht, die über das ‚emg-Diagnose'-Profil verfügen. Ist dies der Fall, wird eine Verbindung mit diesem Diagnose-Server aufgebaut. Hier würden bei Bedarf diverse Sicherheitsmechanismen wie Schlüsselaustausch oder Passwortabfragen aktiviert, die im Labornetzwerk nicht weiter umgesetzt wurden.

Nachdem der Diagnose-Server die Verbindung mit dem entsprechenden Client bestätigt hat, werden vom Diagnose-Client aus die verfügbaren Steuergeräte im Netzwerk abgefragt. In aktuellen Fahrzeugen wird dieser Dienst als das Abfragen der Gateway-Verbauliste bezeichnet. Somit weiß der Diagnose-Client, mit welchen Steuergeräten eine Kommunikation möglich ist. In unserem Beispiel erfährt man so, dass das gesuchte Beifahrer-Steuergerät vorhanden und unter der Adresse ‚2.3.2' erreichbar ist.

Als nächsten Schritt fragt der Diagnose-Client über einen Dienst den Status des Steuergeräts ‚2.3.1' ab. Dafür wird ein Telegramm aufgebaut, das als Zieladresse die ‚2.3.1' und als Kommando den Dienst ‚Status abfragen' enthält. Der Diagnoseserver empfängt dieses Telegramm und leitet es im dynamischen Teil des Kommunikationszyklus weiter. Alle angeschlossenen FlexRay-Knoten empfangen dieses Telegramm vom Diagnose-Server und werten die Zieladresse aus. Anhand ihrer fest eingespeicherten Routingtabellen können sie entscheiden, ob dieses Telegramm für sie relevant ist oder nicht. Sind sie als Gateway für diese Zieladresse zuständig, wird das Telegramm unter einer bestimmten Diagnose-CanID über den angeschlossenen CAN-Bus gesendet. Die CAN-Steuergeräte empfangen die Diagnosenachricht und werten wiederum die Zieladresse aus. Fungieren die CAN-Steuergeräte als Gateway für beispielsweise einen LIN-Bus, müssen dazu ebenfalls Routingtabellen befragt werden. In dem Fall des Beispiels entscheidet das Beifahrer-Steuergerät anhand der Adresse, dass es selbst gemeint ist und baut einen TP-Kanal mit dem Gateway auf. Über diesen schickt es seinen Status (Version, Namen, laufende Applikation usw.) mit dem Routing-Header, der die Telegrammlänge, die Zieladresse für die Diagnose und die SessionID enthält. Über das Gateway wird dieses Telegramm zum Diagnose-Server und anschließend zum Diagnose-Client geschickt, der diese Daten dem Benutzer visualisiert.

6.5.3 Update-Programmierung

Der Diagnose-Client zeigt den, über den Dienst ‚Status abfragen' erhaltenen, Steuergeräte-Status an, zudem auch die Programmversion gehört. Entscheidet man sich zur Programmierung einer neuen Softwareversion, wählt man den Dienst ‚ECU flashen'. Die Diagnoseapplikation fragt dazu nach einem Flash-Container, der auf dem Steuergerät programmiert werden soll. Jetzt erfolgen einige Sicherheitsabfragen nach Authenzität des Codes, der Programmierauthorisierung oder nach Randbedingungen wie das Zulassen nur aktuellerer Versionen, auf die allesamt bei der Realisierung im Labornetzwerk verzichtet wurden.

Der Diagnoseclient kann mit den Informationen über die Größe des Flash-Containers und der Zieladresse ein Telegramm zur Flash-Anfrage aufbauen. Dieses wird analog des ‚Status abfragen'-Telegramms über Diagnose-Server und Gateway zum Steuergerät vermittelt. Das Steuergerät kann nun die Flash-Anfrage negativ oder positiv beantworten. Bei einer negativen Antwort wird der Grund mitgeteilt, während bei einer positiven Antwort eine Zeit dem Gateway gesendet, die das Steuergerät für den Abstieg zum Bootloader-Flash benötigt. Diese Kommunikation und das erneute Aufbauen des TP-Kanals für die Flashdaten finden nur zwischen Gateway und Steuergerät statt. Das Gateway gibt nach außen nur die Antwort des Steuergeräts weiter. Beim Empfang einer positiven Antwort beginnt der Diagnose-Client

anschließend mit der Übertragung des Flash-Containers. Eine Steuerung der Kommunikation durch den Diagnose-Server ist denkbar, wurde aber in der vorliegenden Arbeit nicht umgesetzt.

Analog zur Übertragung der Status- und Flash-Anfragen finden die Daten ihren Weg über den schnellen FlexRay-Backbone zu dem Gateway, an dem der Ziel-Datenbus angeschlossen ist. Das Gateway hat bereits durch die vorherige Flash-Anfrage und die positive Antwort des Steuergeräts dieses veranlasst, die laufende Applikation zu beenden, in den Bootloader-Flash abzusteigen und einen aktiven TP-Kanal aufgebaut. Über diesen Kanal gelangen die Flash-Daten zum Steuergerät und werden dort programmiert. Durch die Mechanismen des Transportprotokolls ist das Steuergerät in der Lage, die Kommunikation mit dem Gateway zu steuern. Dauern die Lösch- und Schreibroutinen für den Flash-Speicher zu lange, besteht die Gefahr eines Pufferüberlaufs. In dem Fall wird dem Gateway signalisiert, die weitere Datenübertragung zu verzögern bis das Steuergerät die erneute Bereitschaft signalisiert.

Bei Beendigung der Programmierung schickt das Steuergerät ein positive Bestätigung zurück zum Diagnose-Client, der darauf hin das Kommando ‚Integrität überprüfen' mit einer Prüfsumme des Flash-Containers sendet. Das Steuergerät berechnet beim Empfang dieser Nachricht eine Prüfsumme seines programmierten Speicherbereichs und vergleicht diese mit der empfangenen Prüfsumme. Treten Unterschiede auf, erhält der Diagnose-Client eine negative Antwort und der Programmiervorgang muss wiederholt werden. Bei einer positiven Überprüfung der Checksummen schreibt das Steuergerät den aktuellen Status, informiert den Diagnose-Client und startet die neue Applikation.

7 Diskussion der Ergebnisse, Zusammenfassung und Ausblick

Durch die vielfältigen Anforderungen der Kunden, des Wettbewerbs und der Gesetzgebung wird der Anteil der Elektronik im Fahrzeug weiter zunehmen und den Komplexitätsgrad zusätzlich erhöhen. Zentraler Bestandteil der Fahrzeugelektronik ist dabei die Vernetzung der einzelnen elektronischen Systeme untereinander. Das Kommunikationssystem eines Fahrzeugs bestimmt im Wesentlichen die Leistungsfähigkeit des elektronischen Gesamtsystems.

Ausgehend von der geschichtlichen Entwicklung der Kraftfahrzeugelektronik ist im Kapitel 2 die Entstehung der Kommunikationsnetzwerke heutiger Fahrzeuge aufgezeigt. Diesen Netzwerken liegt mit der zentralen Gatewayarchitektur eine historisch gewachsene Kommunikationsstruktur zu Grunde, die für zukünftige Anforderungen an ihre Grenzen stößt. Besonders im Bereich der Umfeldsensorik und innovativen Assistenzsysteme, wie sie mit X-by-Wire möglich sind, werden zeitgesteuerte Datenbusse mit hoher Übertragungsrate wie dem FlexRay benötigt. Ein zentrales Super-Gateway, das die Vielzahl an ereignis- und zeitgesteuerten Datenbussen in einem Fahrzeug unter Echtzeitbedingungen bedienen muss, wird nur noch unter hohem technischem und ökonomischem Aufwand realisierbar sein.

In dieser Arbeit wurde mit dem Backbone-Ansatz eine Kommunikationsarchitektur vorgestellt, die zukünftigen Anforderungen gewachsen ist. Durch ihren modularen Aufbau trennt die Backbone-Architektur die einzelnen Domänen voneinander und verringert dadurch das Datenaufkommen innerhalb der Domänen. Jede dieser Domänen kann durch diesen Ansatz als abgeschlossenes und überschaubares System behandelt werden und ist fahrzeugmodellübergreifend einsetzbar. Die Schnittstellen zwischen Domänen und Backbone sind klar definiert und, im Idealfall, standardisiert. Es kommen einfache modulare Gateways zum Einsatz, die eine Entkopplung der verschiedenen zeit- und ereignisgesteuerten Bussysteme bewirken.

Ein wesentlicher Bestandteil solch einer dezentralen Architektur spielt die Schnittstelle nach „außen“. An diese, als Diagnosezugang bezeichnete, Schnittstelle werden mit wachsendem Anteil von Elektronik und Software immer höhere Anforderungen gestellt. Über sie erfolgt die vollständige Diagnose des elektronischen Gesamtsystems eines Fahrzeugs einschließlich der Update-Programmierung von Fahrzeug-Steuergeräten. Der Diagnosezugang muss deshalb sicher und für hohe Datenübertragungsraten ausgelegt sein. Eine zusätzliche Anforderung im Rahmen dieser Arbeit lag in dem drahtlosen Zugang zum Fahrzeug, um die Vorteile durch den Verzicht auf Kabel zu nutzen.

Ein weiterer Schwerpunkt dieser Arbeit bestand in der Untersuchung, wie sich auf Grund der unterschiedlichen Technologien zwischen Masken-ROM und Flashspeicher der Anteil nachträglich programmierbarer Steuergeräte im Fahrzeug entwickeln wird. Als Ergebnis lässt sich festhalten, dass mittelfristig an Steuergeräten mit Flashspeichern kein Weg vorbei führt. Zu groß sind die Vorteile gegenüber dem Masken-ROM, der einzig mit seinem, zumindest bei hohen Stückzahlen, niedrigerem Preis punkten kann. Aber auch hier wird bei zunehmender

Verbreitung im Fahrzeug der Flashspeicher weiter aufholen. Als Konsequenz dieser Entwicklung lässt sich ableiten, dass der Softwareanteil, der nachträglich ins Fahrzeug geflasht werden wird, in hohem Maße steigt. Die gesamte Kommunikationsarchitektur und speziell der Diagnosezugang müssen für diese Anforderungen ausgelegt sein.

Bei den Betrachtungen zum Einsatz von Flashspeichern im Fahrzeug wurden mit dieser Arbeit weitere Anforderungen abgeleitet, die derzeit bei den Herstellern noch nicht genügend berücksichtigt werden. Derzeit befinden sich bei der überwiegenden Anzahl von Steuergeräten mit Flashspeichern das Fahrprogramm und der Bootloader-Flash noch im gleichen Speicherbaustein, so dass ein Überschreiben oder eine Manipulation des Bootloader-Flashs nicht ausgeschlossen werden kann. Hier bietet sich der Einsatz zweier getrennter Speicher für den Bootloader-Flash und das Fahrprogramm an, die für die im Labornetzwerk eingesetzten Steuergeräte nicht zur Verfügung standen. Zumindest sollten die verwendeten Speicherbausteine kleinere Sektorgrößen besitzen, von denen sich einige im Idealfall vor Löschung schützen lassen. Der in dieser Arbeit entwickelte Bootloader-Flash muss durch den gemeinsamen Speicher mit sehr viel Aufwand und über Umwege, speziell bei der Interruptbehandlung, die Trennung mit dem Fahrprogramm sicherstellen. Er kann allerdings als exemplarischer Konzeptvorschlag für einen Bootloader-Flash bei Steuergeräten mit herkömmlichem Speicheraufbau dienen und wurde ebenfalls in anderen Projekten erfolgreich eingesetzt (STEP-X, siehe Abschnitt 2.5.2).

Durch die Möglichkeit der Neuprogrammierung der Steuergerätesoftware lassen sich Hard- und Software nicht mehr als eine Einheit betrachten. Aus diesem Grund müssen die einzelnen Kombinationen zwischen Hard- und Software(versionen) qualifiziert und freigegeben werden, um die Integrität des Systems über den gesamten Lebenszyklus zu gewährleisten. Dies setzt allerdings eine eindeutige Identifizierung sowohl der Hardware als auch der Software voraus. Für eine derartige Anforderung muss der Flashspeicher einen Speicherbereich zur Verfügung stellen, der nur ein einziges Mal programmiert werden kann und in dem beispielsweise eine über den gesamten Lebenszyklus eindeutige Seriennummer abgelegt ist [TEhlers03].

Eine wichtige Forderung beim Einsatz von Flashspeichern ergab sich unmittelbar aus den durchgeführten Untersuchungen mit dem Labornetzwerk zur Update-Programmierung. Bei einer Programmierung des Flash-Speichers muss eine über den gesamten Programmiervorgang konstante Spannung sichergestellt werden, da sonst zu wenig Ladung in die einzelnen Speicherzellen gelangt. Diese können ihren Zustand in dem Fall nur für kurze Zeit speichern, worauf auf eine vermeintlich erfolgreiche Programmierung geschlossen wird. Erst zu einem späteren Zeitpunkt verlieren diese Speicherzellen ihre Ladung und das Verhalten der Steuergerätesoftware wird unvorhersagbar. Daher müssen Mechanismen in den Flashspeicher integriert werden, die eine Programmierspannung überwachen und eine gesicherte Aussage über den Erfolg einer Programmierung treffen können.

Für die Untersuchungen, wie sich die dezentrale Backbone-Architektur gegenüber den an sie gestellten Anforderungen verhält und welche Besonderheiten sich im Vergleich zur zentralen Gateway-Architektur ergeben, wurde ein exemplarisches Fahrzeug-Kommunikationsnetzwerk im Labor aufgebaut. Als Backbone wird der zeitgesteuerte FlexRay eingesetzt, während der drahtlose Diagnosezugang über

Bluetooth realisiert ist. Am Backbone sind zwei CAN-Netzwerke angeschlossen, von denen eines mit einem LIN-Netzwerk verbunden ist. Für diese einzelnen Netzwerke wurden eigene Domänenkommunikationen, auch als Standardkommunikation bezeichnet, umgesetzt. Der Schwerpunkt der Betrachtungen lag bei dieser Arbeit allerdings auf der Diagnosekommunikation und dabei speziell auf der Update-Programmierung von Fahrzeug-Steuergeräten über eine Backbone-Architektur.

Für die Umsetzungen der drahtlosen Diagnoseschnittstelle und der Backbone-Architektur wurden auf bewährte Mechanismen der IT-Welt zurückgegriffen. Die Schnittstelle wird nicht mehr als einfacher „elektrischer" Zugang zum Fahrzeug behandelt, sondern kapselt als eigenes Modul das Fahrzeug vollständig von der Umwelt. Für den Zugriff auf die Steuergeräte stellt die Schnittstelle über eine Client/Server-Architektur Dienste nach „außen" zur Verfügung. Dieses Vorgehen hat den großen Vorteil, dass Zugriffe auf das Fahrzeug nur über definierte Schnittstellen und Zugriffsberechtigungen möglich sind. Gegenüber einem drahtgebundenen Zugang, der nur über eine exklusive Schnittstelle durch einen Stecker verfügt, besitzt die Realisierung mit einem drahtlosen Verfahren weitere Vorteile. So ermöglicht der drahtlose Zugang in Verbindung mit der Client/Server-Architektur parallele Zugriffe verschiedener Clients. Dadurch lässt sich die verfügbare Bandbreite durch einen oder mehrere Clients optimal ausnutzen. Da der Diagnosezugang direkt am Backbone mit seiner hohen Übertragungsrate angeschlossen ist, ist ein schneller Datentransport in das Fahrzeug gewährleistet.

Eines der wichtigsten Ergebnisse dieser Arbeit ist der Nachweis der prinzipiellen Machbarkeit einer Kommunikationsarchitektur, die sich sehr stark an bewährten Architekturen aus der IT-Welt orientiert. Die in der vorliegenden Arbeit herausgestellten Vorteile der Backbone-Architektur im Fahrzeug gegenüber eines Super-Gateways konnten mit Hilfe des Labornetzwerks bestätigt werden. Insbesondere der modulare Aufbau und die damit mögliche klare Trennung der einzelnen Domänen erwiesen sich als sehr hilfreich bei der Auslegung der Kommunikationsstruktur. Gleichzeitig reduzierte sich mit dem Ersetzen des bislang üblichen komplexen Super-Gateways durch kleine einfache (Software)Gateways der Aufwand bezüglich Rechenleistung und Implementierung. Hier zeigte sich der Nutzen des modularen Konzepts, in dem die Software für das FlexRay-CAN-Gateway einmal entwickelt und auf mehreren Gateways eingesetzt werden konnte.

Der Verzicht auf einen zentralen Kommunikationsknoten, wie es ein Super-Gateway darstellt, erlaubt die optimale Ausnutzung der zur Verfügung stehenden Übertragungsraten der einzelnen Datenbusse. Insbesondere bei der Update-Programmierung, wo gesparte Zeit mit eingesparten Kosten korreliert, können dadurch große Datenmengen in kurzer Zeit übertragen werden. Allerdings müssen bei der Programmierung eines Steuergeräts, das an einen vergleichsweise langsamen Datenbus wie einem Low-Speed-CAN angeschlossen ist, die Daten schlussendlich über diesen Flaschenhals transportiert werden. Die Entkopplung zwischen einem schnellen und einem langsameren Datenbus übernimmt das jeweilige Gateway, in dem die Daten dort gepuffert werden. Während bei dem Gateway-Konzept das Super-Gateway die gesamte Datenmenge der unter Umständen mehreren zu programmierenden Steuergeräten zwischenpuffern muss, verteilt sich die Datenlast bei der Backbone-Architektur auf mehrere Gateways. Daher eignet sich die Backbone-Architektur im besonderen Maße für die parallele Programmierung

mehrerer Steuergeräte aus, im idealen Fall, unterschiedlichen Domänen. Durch die Implementierung der Diagnoseschnittstelle als Client/Server-System können mehrere Diagnose-Clients eine Update-Programmierung verschiedener Steuergeräte durchführen. Die in diesem Fall begrenzenden Faktoren sind die maximalen Übertragungsraten der drahtlosen Kommunikationsstrecke und des Backbones. Beide Übertragungsstrecken besitzen aber in der Regel eine signifikant höhere Übertragungsrate als die herkömmliche Diagnoseschnittstelle über K-Line oder CAN.

Dieses Ergebnis konnte mit dem Labornetzwerk allerdings nur begrenzt aussagefähig evaluiert werden. Die zur Verfügung stehenden programmierbaren CAN-Steuergeräte besaßen 256 kByte Flash-Speicher, von denen auf Grund der Bootloader-Flash-Implementierung in Zusammenhang mit der nur sektoren-weisen Löschung des Speichers maximal 192 kByte programmiert werden konnten. Die zu programmierende Applikation besitzt eine Größe von ca. 32 kByte, so dass zu Evaluierungszwecken der Flash-Container künstlich manipuliert wurde, um in den freien Speicherbereichen ein nachvollziehbares Muster zu programmieren. Mit diesem „Kunstgriff“ wurden über das Labornetzwerk Flash-Container mit einer Größe von 192 kByte in die CAN-Domänen programmiert.

Für die Übertragung der Diagnosedaten standen ca. 40 % der vorhandenen Datenrate von 5 MBit/s des Prototypen-FlexRays zur Verfügung, wie im Abschnitt 5.2.3 dargestellt wurde. Die daraus resultierenden 2 MBit/s werden nicht vollständig für die Übertragung von Flash-Daten genutzt. Die verfügbare Datenrate reicht aber auf Grund der begrenzten Größe der Flash-Container selbst für das parallele Flashen zweier Steuergeräte aus, den gesamten Daten-Download bis zu den FlexRay-CAN-Gateway einschließlich Verbindungsaufbau und Diagnosesitzungseröffnung in weniger als 10 Sekunden abzuschließen[35]. Im Vergleich mit Erfahrungswerten aus der Praxis an realen Fahrzeugen, bei denen eine Programmierung ähnlicher Datenmengen bis zu einigen Minuten dauern kann, ist somit eine signifikante Verbesserung und Zeitersparnis festzustellen.

In Verbindung mit dem modularen Aufbau bietet die Backbone-Architektur gegenüber der Gateway-Architektur weitere Vorteile, die allerdings einen gewissen Mehraufwand bei der Implementierung erforderlich machen. Durch seine dezentrale Architektur entfällt die Intelligenz über die Kommunikationsstruktur, die bislang im Super-Gateway vorhanden war. Daher muss eine zusätzliche Protokollschicht neben den bestehenden Transportprotokollen eingeführt werden, die eine Vermittlung der Daten über die Kommunikationsstruktur sicherstellt. In dieser Arbeit wurde der Ansatz eines derartigen Routingprotokolls vorgestellt, das sich sehr stark am bewährten Internetprotokoll IP orientiert. Ein wichtiger Baustein bestand in der Festlegung einer fahrzeugweit eindeutigen Adressierung. Die Adresse sollte für das Routing eine Aussage über die Struktur des Systems zulassen und gleichzeitig eine ähnlich hohe Flexibilität wie IP besitzen. Die große Herausforderung liegt in dem Fall in der zehn- bis hundertmal geringeren Übertragungsrate automobiler Datenbusse im Vergleich zu Ethernet und den daraus resultierenden Nachrichtengrößen.

[35] Diese Zeit bezieht sich auf den Download der Daten in das Fahrzeug. Die eigentliche Flashdauer wird immer durch die eingesetzten Flashspeicher und die Übertragungsrate des Datenbusses, an dem das Steuergerät angeschlossen ist, bestimmt.

Der in dieser Arbeit vorgestellte Ansatz eines automobilen Routingprotokolls kommt mit nur zwei Adressbytes aus und erfüllt die gestellten Anforderungen, von denen einige im Widerspruch zueinander stehen, sehr gut. Bei der Bewertung des Protokolls müssen die beiden Kommunikationsarten, Standard- und Diagnosekommunikation, allerdings getrennt voneinander betrachtet werden. Gerade für kleine Datenmengen im Bereich einiger weniger Bytes, wie sie für die Standardkommunikation typisch sind, erzeugt das Routingprotokoll mit seinen Adressbytes sehr viel Overhead. In dem Fall macht ein Routingprotokoll wenig Sinn, da die Kommunikationsteilnehmer und Datenmengen im Vorfeld bereits feststehen und somit auf die Flexibilität eines Routingprotokolls verzichtet werden kann. Für die Standardkommunikation bietet sich aus diesem Grund auch für eine Backbone-Architektur beim Routing der herkömmliche Weg über Verbindungsmatrizen in den einzelnen Gateways an.

Dagegen besteht bei der Diagnosekommunikation eine der Hauptanforderungen in der Erreichbarkeit eines jeden diagnosefähigen Steuergeräts von „außen“ und gerade hier ist die Flexibilität eines Routingprotokolls unabdingbar. Denn bei der Diagnose müssen die Diagnosenachrichten jedes diagnosefähige Steuergerät erreichen können, unabhängig vom Datenbus, an den es angeschlossen ist. Dabei muss die Information über den Absender einer Nachricht bei der Backbone-Architektur innerhalb der Nachricht übermittelt werden, da hier keine zentrale Instanz wie ein Super-Gateway existiert, dass diese Information bereitstellt. Mit der Beschränkung des Routings auf die Diagnosekommunikation wird erst die relative geringe Größe von acht Bytes für die Routing- und Telegramminformationen möglich. Dadurch lassen sich die Adressbytes für die Absenderadresse einsparen, in dem die Absenderadresse innerhalb der zwei (Ziel)Adressbytes verschlüsselt wird. Die verbleibenden Datenbytes werden zur Nachbildung der Transportprotokollmechanismen verwendet.

Mit Hilfe des Labornetzwerks wurde die Funktion des vorgestellten Routingprotokolls innerhalb einer Backbone-Architektur nachgewiesen. Die notwendige Beschränkung auf die Diagnosekommunikation ist durch die gewonnene hohe Flexibilität akzeptabel. Im Zusammenspiel mit dem schnellen Backbone konnten parallele Update-Programmierungen durchgeführt und damit die Vorteile der höheren Datenübertragungsraten optimal ausgenutzt werden. Auf Grund der begrenzten Datenmengen, die im Labornetzwerk programmiert werden können, sind die Ergebnisse nur als ein Indiz für diesen Ansatz zu werten. Für allgemeingültigere Aussagen muss dieser Ansatz mit Hilfe von seriennahen Steuergeräten mit größeren Datenspeichern weiter evaluiert werden. Darüber hinaus sind weitergehende Untersuchungen hinsichtlich der Gatewaysoftware notwendig, so findet beispielsweise in der hier gezeigten Version nur eine rudimentäre TimeOut-Überwachung statt. Bei einer weiteren Verfolgung des Ansatzes muss zusätzlich die Standardkommunikation näher betrachtet werden, die im Vergleich zur hier vorgestellten Realisierung ebenfalls Optimierungspotenzial bietet.

Die weitere Entwicklung des Kommunikationssytems im Fahrzeug wird sich immer mehr in Richtung Backbone-Architektur verlagern. Gleichzeitig verändert sich dadurch das Datenmanagement. Wurden Kommunikationsstrukturen bislang überwiegend in Listen statisch manuell gepflegt, so werden zukünftig mehr Flexibilität und eine stärkere Werkzeug-Unterstützung eine Rolle spielen. Die ersten Ergebnisse des AUTOSAR-Konsortium zeigen ebenfalls in diese Richtung. Es wird immer mehr

die Steuergeräte-Sicht zugunsten einer funktionalen Sicht aufgegeben, die nicht mehr mit physikalischen Daten, sondern mit logischen Signalen arbeitet. Ein Kommunikationssystem hat dann die Aufgabe, die Vernetzung dieser logischen Signale netzwerkübergreifend sicherzustellen. Es wird „autonom“ und von „außen“ transparent arbeiten. Eine Voraussetzung dafür ist ein Routingprotokoll, wie es in dieser Arbeit vorgestellt wurde. Zukünftig wird das Routingprotokoll nur noch ein Teil einer automobilen Middleware sein, die die Netzwerkorganisation übernimmt. Über diese Middleware lassen sich dann Netzwerkstruktur, Kommunikationspartner und Datendefinitionen automatisiert ermitteln.

Für die zu erwartenden Anforderungen bietet die Backbone-Architektur durch ihren modularen Aufbau die größeren Vorteile gegenüber einer Gateway-Architektur. Die notwendigen Eigenschaften an den Backbone sind mit hoher Flexibilität, Echtzeitfähigkeit und hoher Datenübertragungsrate definiert. Allerdings ist man bei der Wahl des Backbones nicht auf FlexRay festgelegt. Auch andere Bussysteme wie IDB1394 oder Ethernet erfüllen diese Anforderungen und können die Aufgaben des FlexRays als Backbone übernehmen. Besonders Ethernet wird durch seine starke Verbreitung in der IT- und Consumer-Elektroinik eine Zukunft im Automobil besitzen. Es bietet bei geringen Kosten eine Vielfach höhere Datenübertragungsrate und einen bereits etablierten Standard. Die Anpassungen von Ethernet an automobil-spezifische Anforderungen wie spezielle Temperatur- und EMV-Bedingungen spielen bei den zu erwartenden Vorteilen nur eine untergeordnete Rolle. In der industriellen Automatisierungstechnik finden (echtzeitfähiges) Ethernet und draht-lose Kommunikation bereits breite Anwendungen, von deren Erfahrungen die Automobilindustrie profitieren kann.

8 Abbildungsverzeichnis

9 Tabellenverzeichnis

10 Literatur

[Adolfsson00] Adolfsson, N., Mattson, J.: „Benchmarking Real-Time-Operating Systems for Automotive Use“
Diplomarbeit, Department of Computer Engineering, Chalmers University of Technology, Göteborg 2000

[Angele05] Angele, G.: „Netzwerke auf Rädern“
Zeitschrift *Elektronik automotive*, Ausgabe 2/2005

[Auto01] Magazin „Automobilelektronik“ – 03/2003, Verlag Moderne Industrie

[AUTOSAR] Automotive Open System Architecture – AUTOSAR
http://www.autosar.org

[Beckmann02] Beckmann, G.: „Ein Hochgeschwindigkeits-Kommunikationssystem für die industrielle Automation“
Dissertation, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, 2002, ISBN 3-98091818-1-X

[Bernhart04] Bernhart, W.: „Software Update und Upgrade - Status der Umsetzung bei Pkw-und Nfz-Herstellern“
5. Euroforum Konferenz „Software im Automobil“, Stuttgart 2004

[BlueZ] BlueZ – Freier Bluetooth Stack für Linux
http://www.bluez.org

[Bosch02] Robert Bosch GmbH: „Autoelektrik / Autoelektronik“
4.Auflage, Vieweg Verlag, Braunschweig/Wiesbaden, 2002,
ISBN 3-52813872-6

[Braess01] Braess H. H., Seiffert U.: „Handbuch Kraftfahrzeugtechnik“
Vieweg Verlag, Braunschweig/Wiesbaden, 2001, ISBN 3-52813114-4

[Broy98] Broy, M., von der Beeck, M., Krüger, I.: „SOFTBED: Problemanalyse für ein Großverbundprojekt: Systemtechnik Automobil – Software für eingebettete Systeme“
BMBF-Abschlussbericht SOFTBED, TU München, 1998

[Busch1] Busch, R.: „Elektronik im Automobil“, Band 1, „Grundlagen und Technik der Halbleiter“
Krafthand Verlag Walter Schulz, Bad Wörishofen, o.J.

[Busch2] Busch, R.: „Elektronik im Automobil“, Band 2, „Halbleiter in der elektronischen Ausrüstung“
Krafthand Verlag Walter Schulz, Bad Wörishofen, o.J.

[CAN] Controller Area Network – CAN
http://www.can-cia.org

[Chodura04] Chodura, H., Hofmann, P.-M., Kalusche, B., Knoblach, J., Spohr, J., Weber, T.: „Herstellerinitiative Software“
Zeitschrift *Elektronik automotive*, Ausgabe 4/2004

[CT2605] Zivadinovic, D.: „Fliegender Wechsel“
Zeitschrift *c't*, Ausgabe 26/2005, S.52

[DCH] Daimler Chrysler Heritage
http://www.daimlerchrysler.com (Sitemap → Heritage)

[Debian] Debian Linux
http://www.debian.org

[Decomsys] DECOMSYS – Dependable Computer Systems GmbH
http://www.decomsys.com

[Detering04] Detering, S.: „Entwicklung eines automobilen CAN-Netzwerkes auf Basis der VW – Standardsoftware“
Studienarbeit, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig, 2004

[Dudenhöff.03] Dudenhöffer, F.: „Trends und Entwicklungen in der Automobil- und Zulieferindustrie“
2.SESAMES-Messe, Emden 2003

[Ellinger04] Ellinger, T.: „Überblick über die Diagnosesysteme bei Volkswagen“
Schulungsunterlagen der IAV GmbH, Gifhorn 2004

[Engine04] Yaegashi, T.: „Combined charge – Award 2004“
Zeitschrift *engine technology international*, Ausgabe 09/2004

[Etschberger94] Etschberger, K.: „CAN Controller Area Network“
Carl Hanser Verlag, 1994, ISBN 3-44617596-2

[Färber95] Färber, G.: „Feldbussysteme“
Oberseminar Prozeßrechentechnik, Lehrstuhl für Prozeßrechner, TU München, WS 1994/1995

[Faulbacher03] Faulbacher, M., Wagner, G.: „Updateprogrammierung entlang der Prozesskette aus Sicht eines Automobilherstellers“
IIR-Konferenz „Software-Flashen im Kfz“, 2003

[Fennel05] Fennel, H., Judaschke, U.: „20 Jahre softwarebasierte Mikroelektronik im Dienst der Fahrzeugsicherheit“
ATZ/MTZ-Sonderheft „*Automotive Electronics*“, März 2005

[FlexRay] FlexRay
http://www.flexray.com

[FlexRay05] FlexRay-Konsortium, FlexRay Spezifikation Version 2.1, 2005

[Förster84] Förster, H.J.: „Wünsche des Automobilkonstrukteurs an die Elektronik“
1. VDI-Tagung „Elektronik im Kraftfahrzeug“, Baden-Baden, 1984
VDI-Berichte 515, VDI-Verlag, Düsseldorf 1984, ISBN 3-18090515-8

[Franz04] Franz, W.: „Car-to-Car Communication – Anwendungen und aktuelle Forschungsprogramme in Europa, USA und Japan“
Fachtagungsbericht GMM, VDE-Kongress Berlin 2004

[Gagalski05] Gagalski, P.: „Entwicklung eines FlexRay-CAN-Gateways auf Basis eines DECOMSYS FlexRay-Clusters“
Studienarbeit, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig, 2005

[Grzemba04] Grzemba, A.: „Workshop-LIN-Bus“
Designer&Elektronik Entwicklerforum, Ludwigsbur 2004

[Habiger98] Habiger, E.: „Elektromagnetische Verträglichkeit: Grundzüge ihrer Sicherstellung in der Geräte- und Anlagentechnik“
3. überarbeitete und erweiterte Auflage, Hüthig Verlag, Heidelberg 1998, ISBN 3-77852645-6

[Heigl04] Heigl, H.-P.: „Flashtechnologie im Kraftfahrzeug – Status und Ausblick“
IIR-Konferenz „Software-Flashen im Kfz“, Stuttgart 2004

[Heinze79] Heinze, G.W.: „Verkehr schafft Verkehr – Ansätze zu einer Theorie des Verkehrswachstums als Selbstinduktion“
Berichte zur Raumforschung und Raumplanung, Jg. 23 Heft 4/5, Wien 1979

[Hentze01] Hentze, J., Heinecke, A., Kammel, A.: „Allgemeine Betriebswirtschaftslehre“
UTB-Verlag, Stuttgart 2001, ISBN 3-82522040-0

[Herbst06] Herbst, U.: „Langfristige Technologieabschätzung für spurgebundene Verkehrssysteme“
Dissertation, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, geplant 2006, ISBN

[Herrmann03] Herrmann, W.: „Software-Update für BMW-Fahrzeuge“
IIR-Konferenz „Software-Flashen im Kfz“, Stuttgart 2003

[HIS] Herstellerinitiative Software – HIS
http://www.automotive-his.de

[Hofman01] Hofmann, P., Thurner, T.: „Neue Elektrik/Elektronik Architekturansätze“
Elektronik im Kraftfahrzeug, Baden-Baden 2001
VDI-Berichte 1646, VDI-Verlag, Düsseldorf 2001, ISBN 3-18091646-X

[Högl03] Högl, H.: „Grundlagen und Anwendungen von Flash-Speichern“
IIR-Konferenz „Software-Flashen im Kfz“, Stuttgart 2003

[Hoika02] Hoika, J., Preuschoff, C.: „Chip-Intelligenz für automobile Antriebe“
Zeitschrift *Elektronik automotive*, Ausgabe 12/2002

[Huber03] Huber, M., Weber, T., Miehling, T.: „Software-Update, Variantenbildung und Feldprogrammierung für individualisierte Fahrzeuge auf Basis eines intelligenten Datenkonfigurationssystems“
Tagung „Elektronik im Kraftfahrzeug“, Baden-Baden 2003
VDI-Berichte 1789, VDI-Verlag, Düsseldorf 2003, ISBN 3-18091789-X

[Huber04] Huber, R.: „Informationstechnik im Automobil - Vom einfachen Bussystem zum anspruchsvollen Datennetz“
Zeitschrift *automotive electronics + systems*, Ausgabe 5-6/2004

[IDB] Intelligent Transportation System Data Bus – IDB
http://www.idbforum.org

[ifmo02] Institut für Mobilitätsforschung (ifmo): „Zukunft der Mobilität – Szenarien für das Jahr 2020“
http://www.ifmo.de

[ifmo05] Institut für Mobilitätsforschung (ifmo): „Zukunft der Mobilität – Szenarien für das Jahr 2025“
1.Auflage, Verlag BMW AG, 2005, ISBN 3-93216926-3

[IKF04] Varchmin, J.-U.: „Industrielle Kommunikation mit Feldbussen“
Vorlesungsskript SS 2004, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig 2004

[Infineon03] Infineon Technologies (Hrsg.): „Halbleiter technische Erläuterungen, Technologien und Kenndaten“
3. Auflage, Publics Corporate Publishing, 2003, ISBN 3-89578205-X

[Intel02] Intel: „Datasheet: 3V Intele Strata Flash Memory“, Stand: 2004
Internet:
http://web.mit.edu/6.111/www/s204/newkirt/datasheets/28f128j3A.pdf

[Intel05] Intel: „Moore’s Law – Overview“, Stand: 2005
Internet: http://www.intel.com/technology/silicon/mooreslaw/index.htm

[Internet1] Internet: http://www.kraftfahrzeugtechnik-heute.de
Übersicht über die Bosch-Kraftfahrzeugtechnikentwicklung

[INVENT] Intelligenter Verkehr und nutzergerechte Technik – INVENT
http://www.invent-online.de

[ISO15765] ISO-Standard 15765: „Road vehicles –Diagnostivs on Controller Area Network (CAN) – Part 2: Network layer services” – (ISO-TP)

[Kaindl03] Kaindl, M.: „Processor Strategy Criteria“
BMW – Forschungsbericht, BMW AG, EI-6-H, 2003

[KEhlers86] Ehlers, K.: „Automobilelektronik – Start gelungen, wie geht es weiter? Strategische Überlegungen für die nächste Dekade“
2. VDI-Tagung „Elektronik im Kraftfahrzeug“, Baden-Baden, 1986
VDI-Berichte 612, VDI-Verlag, Düsseldorf, 1986, ISBN 3-18090612-X

[Kiencke00] Kiencke, U.: „Ereignisdiskrete Systeme. Modellierung und Steuerung verteilter Systeme“
Verlag Oldenbourg, 2000, ISBN 3-48624150-8

[Kifmann02] Kifmann, A.: „Die Softwarefalle“
Editorial *Auto & Elektronik*, Mai 2002

[Kopetz97] Kopetz, H.: „Real-Time Systems Design Principles for Distributed Embedded Applications”
Kluwer Academic Publishers, 1997, ISBN 0-79239894-7

[Kozlowski04] Kozlowski, F.: „Elektromaschinen im Antriebsstrang: Spaß am Sparen“
Euroforum Konferenz „Elektroniksysteme im Automobil“, München 2004

[Kruse04] Kruse, J.: „Pleiten, Pannen, Perspektiven“
Zeitschrift *Automobilwoche*, Ausgabe 11/2004

[Lange01] Lange, K., Bortolazzi, J., Marx, D., Wagner, G., Gresser, K.: „Hersteller-Initiative Software“
Elektronik im Kraftfahrzeug, Baden-Baden 2001
VDI-Berichte 1646, VDI-Verlag, Düsseldorf 2001, ISBN 3-18091646-X

[Lange03] Lange, K.: „Einführung in die Problemstellung der Elektronikentwicklung im Automobilbereich“
GZVB-Tagung, Braunschweig 2003

[Lay94] Lay, G.M.: „Die Geschichte der Straße“
2. Auflage, Campus Verlag, Frankfurt/Main, New York, 1994,
ISBN 3-59335132-3

[Leitner04] Leitner, R.: „Flashen am Beispiel des Audi A8“
IIR-Konferenz „Flashtechnologie im Kfz“, Stuttgart 2004

[Leonhard01] Leonhard, R.: „Verbrauchs- und Abgaskonzepte zukünftiger Ottomotoren“
55. Internationales Motorpressekolloquium, Boxberg, 2001

[LIN] Local Interconnect Network – LIN
http://www.lin-subbus.org

[LIN04] LIN-Konsortium, LIN Spezifikation Version 2.0, 2004

[Loch04] Loch, M.: „Embedded Flash für Automobilanwendungen“
IIR-Konferenz „Flashtechnologie im Kfz“, Stuttgart 2004

[Lübke04] Lübke, A.: „Car-to-Car Communication – Technologische Herausforderungen“
Fachtagungsbericht GMM, VDE-Kongress Berlin 2004

[Lukas04] Lukas, W.-D.: „Neue Technologien erfordern neue Wege der Qualifikation“
ISST-Forum, Berlin 2004

[MAT04] Varchmin, J.-U.: „Mikrorechner in der Automatisierungstechnik“
Vorlesungsskript SS 2004, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig 2004

[Mercer04] Studie Mercer Management
Zeitschrift *Automobil Elektronik*, Ausgabe 2/2004, Verlag Moderne Industrie

[MOST] Media Oriented Systems Transport – MOST
http://www.mostcooperation.com

[Müller03] Müller, S.: „Software-Update von Steuergeräten in der Vertriebsorganisation“
IIR-Konferenz „Software-Flashen im Kfz“, Stuttgart 2003

[Münster99] Münster, R.: „Neue Flash-Technologien für’s Kfz: Need for Speed“
Zeitschrift *Auto & Elektronik*, Ausgabe 1/1999

[Neumann04] Neumann, K.-Th.: „Prozesse für die Steuergeräteentwicklung im Volkswagen-Konzern“
Automobil Elektronik Internationaler Fachkongress 2004, Fortschritte in der Automobil-Elektronik, Ludwigsburg 2004

[Pasemann03] Pasemann, K.: „Bussysteme und Instandsetzung“
BAIS-Tagung, Salzburg 2003

[Petersen99] Petersen, R.: „Mobilität für morgen“
Tagung „Autofrei leben“, Weimar 1999

[Qt] Qt – plattformübergreifendes GUI Entwicklungsframework
http://www.trolltech.com/products/qt

[Ranft03] Ranft, S.: „Autoindustrie etabliert offenen Standard“
Zeitschrift *Elektronik automotive*, Ausgabe 5/2003

[Rech04] Rech B.: „Die Elektrik und Elektronik eines modernen Automobils“
Fachtagungsbericht GMM, VDE-Kongress, Berlin 2004

[Reichart04] Reichart, G.: „Systemarchitektur im Kraftfahrzeug“
8. Euroforum Jahrestagung „Elektronik im Automobil“, München 2004

[Reichart05] Reichart, G.: „Zukünftige Systemarchitekturen im Kraftfahrzeug“
25. Haus der Technik-Tagung „Elektronik im Kraftfahrzeug“, München 2005

[RFC2460] RFC 2460: „Internet Protocol, Version 6 (IPv6) Specification“
Internet Engineering Task Force, 1998
http://www.ietf.org

[RFC791] RFC 791: „Internet Protocol“
Internet Engineering Task Force, 1981
http://www.ietf.org

[RFC826] RFC 826: „Address Resolution Protocol“
Internet Engineering Task Force, 1982
http://www.ietf.org

[Ringler02] Ringler, T.: „Entwicklung und Analyse zeitgesteuerter Systeme“
Dissertation, Institut für Automatisierungs- und Softwaretechnik, Universität Stuttgart, Shaker Verlag, Aachen 2002, ISBN 3-83220782-1

[Rogers95] Rogers, E.M.: „Diffusion of innovations“
Free Press, New York 1995, ISBN 0-02926671-8

[RSAP] European Road Safety Action Programme
http://europa.eu.int/comm/transport/road/roadsafety/rsap/index_en.htm

[RTAI] Real Time Application Interface
http://www.rtai.org

[Schade05] Schade, R., Varchmin, J.-U.: „Ansatz einer Datenbusarchitektur auf Basis von FlexRay“
25. Haus der Technik-Tagung „Elektronik im Kraftfahrzeug“,
München 2005

[SchadeF04] Schade, F.: „Konzeptionierung und Realisierung eines LIN Netzwerkes in Hard- und Software“
Diplomarbeit, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig, 2004

[Scharnhorst04] Scharnhost, T.: „Herausforderungen für die Einführung neuer Elektronik-architekturen im Fahrzeug“
2.ISST-Forum, Berlin, 2004

[Schäuffele03] Schäuffele, J., Zurawka, T.: „Automotive Software Engineering“
1.Auflage, Vieweg Verlag, Wiesbaden, 2003, ISBN 3-52801040-1

[Schleuter02] Schleuter, W.: „Herausforderungen in der Automobil-Elektronik“
IKB Unternehmerforum, Köln 2002

[Schrey02] Schrey, U.: „Die Bedeutung der Software in der Automobilelektronik“
3. Euroforum Konferenz „Software im Automobil“, Stuttgart 2002

[Schumpeter12] Schumpeter, J.A.: „Theorie der wirtschaftlichen Entwicklung“
Faks. d. Erstausg. 1912, Verlag Duncker Humblot, Berlin 1997,
ISBN 3-42807725-3

[Schwab00] Schwab, A.: „Innovationen durch Elektronik“
ATZ/MTZ-Sonderheft „*Automotive Electronics*“, Januar 2000

[Schwarz78] Schwarz, H.: „Elektronik im Kraftfahrzeug“
VDE-Verlag, Berlin, 1978, ISBN 3-8007-1152-4

[Seiffert05] Seiffert, U., Varchmin J.-U.: „Der Nutzen von mehr Fahrzeugelektronik“
ATZ/MTZ-Sonderheft „*Automotive Electronics*“, März 2005

[Siemers02] Siemers, C., Sikora, A.: „Taschenbuch Digitaltechnik“
Hanser Fachbuch Verlag, 2002, ISBN 3-44621862-9

[Specht05] Specht, B.: „Client/Server-Programmierung über Bluetooth zur Realisierung eines Diagnosetools für heterogene Kraftfahrzeug-Kommunikationsnetz-werke“
Studienarbeit, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig, 2005

[Specks01] Specks, W.: „Multimediale Information, Kommunikation und Unterhaltung im Automobil“
DGON Jahreshauptveranstaltung 2001 „Mobilität und Sicherheit“, Wolfsburg 2001

[Specks05] Specks, W., Rech, B.: „Car-to-X Communication“, Tagung „Elektronik im Kraftfahrzeug“
Baden-Baden 2005, VDI-Berichte 1907, VDI-Verlag, Düsseldorf 2005, ISBN 3-18091907-8

[Spika02] Spika, M.: „Bluetooth-Integration in RTAI-Linux und Aufbau einer drahtlosen Kommunikation zwischen zwei PCs (Windows/Linux)“
Studienarbeit, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig, 2002

[Tauscher04] Tauscher, J.: „Anforderungen an den Standard Flashloader für Kfz-Steuergeräte“
IIR-Konferenz „Flashtechnologie im Kfz“, 2004

[Teepe00] Teepe, G.: „Maßnahmen zur Steigerung der Innovationsgeschwindigkeit in der Automobilelektronik, ein Vergleich mit der Computerindustrie“
Tagung „Elektronik im Kraftfahrzeug“, Baden-Baden 2000
VDI-Berichte 1547, VDI-Verlag, Düsseldorf 2000, ISBN 3-18091547-1

[Teepe04] Teepe, G.: „Flashtechnologie im Kraftfahrzeug – Status und Ausblick“
IIR-Konferenz „Flashtechnologie im Kfz“, Stuttgart 2004

[TEhlers03] Ehlers, T.: „Verfahren zur Sicherstellung der Systemintegrität in Fahrzeugen mit vernetzten Steuergeräten“
Dissertation, Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik, TU Braunschweig, Braunschweig 2003,
ISBN 3-98091818-6-0

[Temple05] Temple, C.: „Getting FlexRay on the Road“
3. Euroforum Konferenz „Datenkommunikation im Automobil“,
Heidelberg 2005

[Thoma84] Thoma, P., Vondracek, P.: „Anforderungen an Daten-Bussysteme von Steuergeräten im Kfz und deren Realisierung“
1. VDI-Tagung „Elektronik im Kraftfahrzeug“, Baden-Baden, 1984
VDI-Berichte 515, VDI-Verlag, Düsseldorf 1984, ISBN 3-18090515-8

[Torney04] Torney, M.: „Evaluierung der technischen Anforderungen und Technologien von Flash-Speichern im Kraftfahrzeug sowie Wirtschaftlichkeitsbetrachtungen anhand verschiedener Bewertungs- und Entscheidungsverfahren“
Studienarbeit, Institut für Wirtschaftswissenschaften, Abt. Volkswirtschaftslehre, TU Braunschweig, Braunschweig, 2004

[TP2.0] VW-Konzernlastenheft: „CAN-Transportprotokoll TP2.0“
Version 1.0

[TTP] Time Triggered Protocol
http://www.ttpforum.org

[Varchmin05] Varchmin, J.-U., Schade, R.: „Der Nutzen von Datenkommunikation im Automobil“
3. Euroforum Konferenz „Datenkommunikation im Automobil“,
Heidelberg 2005

[Vaßen05] Vaßen, H.-W.: „Gateway-Strategien“
3. Euroforum Konferenz „Datenkommunikation im Automobil“,
Heidelberg 2005

[Vector] Vector Informatik GmbH
http://www.vector-informatik.de

[WBCSD04] World Business Council for Sustainable Development: „Mobilität 2030 – Die Herausforderungen der Nachhaltigkeit meistern”
Sustainable Mobility Project, Executive Summer 2004
http://www.wbcsd.org/web/mobilitypubs.htm

[Winterkorn05] Winterkorn, M.: „Innovationen für den Kunden“
Zeitschrift *Elektronik automotive*, Ausgabe 4/2005

[Wollert02] Wollert, J.: „Das Bluetooth Handbuch“
Franzis Verlag, 2002, ISBN 3-77235323-1

[Zimmerm.06] Zimmermann, W., Schmidgall, R.: „Bussysteme in der Fahrzeugtechnik“
Vieweg Verlag, 1. Auflage, Wiesbaden, 2006, ISBN 3-83480166-6

11 Lebenslauf

Name	Ralf Schade
Geburtsdatum/-ort	02.04.1974 in Erfurt
Familienstand	ledig

Schulausbildung

1980 – 1990	Polytechnische Oberschule 25, Erfurt
1990 – 1992	Königin-Luisen-Gymnasium, Erfurt

Hochschulausbildung

1992 – 1999	Studium der Elektrotechnik an der TU Braunschweig Schwerpunkt Mess-, Regelungs- und Automatisierungstechnik

Wehrdienst

1999 – 2000	Kommando 3. Luftwaffendivision, Berlin

Tätigkeiten

2000 – 2006	Wissenschaftlicher Mitarbeiter am Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik TU Braunschweig
seit 2006	Projektingenieur bei der IAV GmbH, Gifhorn